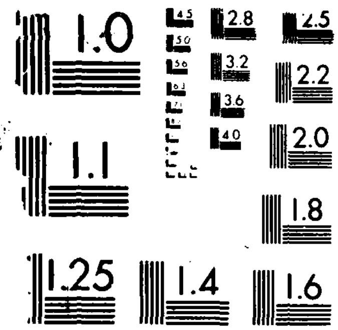
AD-A186 587 SOFTWARE DATA BASE DEVELOPMENT VOLUME 1(U) ANALYTIC SCIENCES CORP READING MA A C NAJBERG 25 JUN 84 TR-4612-5-1-VOL-1 ESD-TR-87-166-VOL-1 1/2

UNCLASSIFIED F33657-82-D-0253/0014 F/G 12/5 NL



MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS 1963-A

12

ESD-TR-87-166 TR-4612-5-1

Software Data Base Development
Volume I

ANDREW C. NAJBERG

The Analytic Sciences Corporation
One Jacob Way
Reading, Massachusetts 01867

25 June 1984

APPROVED FOR PUBLIC RELEASE: DISTRIBUTION UNLIMITED

Prepared For

ELECTRONIC SYSTEMS DIVISION
AIR FORCE SYSTEMS COMMAND
DEPUTY COMPTROLLER
HANSCOM AIR FORCE BASE, MASSACHUSETTS 01731

LEGAL NOTICE

OTHER NOTICES

" THIS TECHNICAL REPORT HAS BEEN REVIEWED AND IS APPROVED FOR PUBLICATION."

JOSEPH P. DEAN, Capt, USAF
Senior Software Cost-Research Analyst
Directorate of Cost
Comptroller

ELLEN M. COAKLEY
Technical Director of Cost
Comptroller

Ellen M. Coakley

FOR THE COMMANDER

DAVID G. KANTER, Colonel, USAF
Comptroller

D G Kanter

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION Unclassified	1b. RESTRICTIVE MARKINGS
2a. SECURITY CLASSIFICATION AUTHORITY	3. DISTRIBUTION / AVAILABILITY OF REPORT Approved for Public Release; Distribution Unlimited
2b. DECLASSIFICATION / DOWNGRADING SCHEDULE	
4. PERFORMING ORGANIZATION REPORT NUMBER(S) TR-4612-5-1	5. MONITORING ORGANIZATION REPORT NUMBER(S) ESD-TR-87-166
6a. NAME OF PERFORMING ORGANIZATION The Analytic Sciences Corp. 6b. OFFICE SYMBOL (If applicable)	7a. NAME OF MONITORING ORGANIZATION ELECTRONIC SYSTEMS DIVISION (ACCR)
6c. ADDRESS (City, State, and ZIP Code) One Jacob Way Reading, Massachusetts 01867	7b. ADDRESS (City, State, and ZIP Code) Hanscom AFB Massachusetts, 01731-5000
8a. NAME OF FUNDING / SPONSORING ORGANIZATION Deputy Comptroller 8b. OFFICE SYMBOL (If applicable) ESD/ACCR	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER F33657-82-D-0253/0014
8c. ADDRESS (City, State, and ZIP Code) Hanscom AFB Massachusetts, 01731-5000	10. SOURCE OF FUNDING NUMBERS PROGRAM ELEMENT NO. / PROJECT NO. / TASK NO. / WORK UNIT ACCESSION NO.

11. TITLE (Include Security Classification)
Software Data Base Development Volume I

12. PERSONAL AUTHOR(S)
Andrew C. Najberg

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT (Year, Month, Day)	15. PAGE COUNT
Technical	FROM ____ TO ____	1984 June 25	114

16. SUPPLEMENTARY NOTATION

17. COSATI CODES			18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
FIELD	GROUP	SUB-GROUP	Software Data Collection

19. ABSTRACT (Continue on reverse if necessary and identify by block number)

This document identifies the data elements that need to be collected and methodologies to be used when developing a Software Cost Database. It presents various data collection formats that can be used for data collection. It identifies the various DIDs that should be modified to enhance the software data collection effort. It also maps the various data elements from the data collection formats to several existing software models including COCOMO, PRICE-S, SLIM and JS II.

20. DISTRIBUTION / AVAILABILITY OF ABSTRACT ☐ UNCLASSIFIED/UNLIMITED ☒ SAME AS RPT ☐ DTIC USERS	21. ABSTRACT SECURITY CLASSIFICATION Unclassified
22a. NAME OF RESPONSIBLE INDIVIDUAL Joseph Dean, Captain, USAF	22b. TELEPHONE (Include Area Code) (617) 377-2679 22c. OFFICE SYMBOL ESD/ACCR

DD FORM 1473, 84 MAR — 83 APR edition may be used until exhausted. All other editions are obsolete.

SECURITY CLASSIFICATION OF THIS PAGE

Unclassified

TABLE OF CONTENTS

TABLE OF CONTENTS (Continued)

1. INTRODUCTION

The increasing contribution of software development and maintenance costs to the overall life-cycle cost of DoD weapon systems has been well documented in recent years. In particular, software life-cycle costs are predicted to be in excess of 80% of total computer hardware/software system life-cycle cost and in excess of 50% of the total system costs by the end of the decade (Ref. 1). This situation reflects the decreasing costs of computer hardware and the expanded use of embedded computers in DoD systems because of their functional capability. Recognition of this accelerating shift of cost drivers from hardware to software has resulted in significantly more attention being given to methods of deriving estimates of the resources that will be expended on the software subsystems. Therefore, a method of estimating software cost and schedule requirements with a reasonable confidence level is required to enhance existing management tools.

There currently exist several software cost estimating systems and models which have gained some degree of acceptance within the software cost estimating community. The most commonly used of these are SLIM (Ref. 2), JS-1 (Ref. 3), PRICE-S (Ref. 4), and COCOMO (Ref. 5). However, before any of these models can be used to develop a software cost estimate, they should be validated for use for a particular class of applications. Moreover, if validated, these models must then be calibrated for a particular development environment. A data base which is representative of Air Force Electronic Systems Division (ESD) software development is necessary to make these models useful.

An alternative to the use of the existing software cost models is the development of a unique model which is specifically tailored to the ESD development environment and product characteristics. This approach can overcome several shortcomings of the existing models, in particular, the lack of a:

- Valid method for estimating the cost of software modification or maintenance
- Valid technique for timephasing manpower requirements when resource constraints exist
- Statistical basis for establishing confidence intervals for an estimate
- Capability to estimate software size.

Development of this unique model requires a comprehensive data base which contains software characteristics and parameters for development projects that reflect the development environment and applications of ESD.

1.1 PURPOSE

The main objective of this effort is to develop a software cost data base which can support the cost estimating process. The ability to use the data base for validation and calibration of the COCOMO, SLIM, JS-1, and PRICE-S software cost models is a major consideration. Since the state-of-the-art of software cost estimating is advancing, the data base must also be flexible enough to be used for model development or enhancement. Finally, the data base should support development of software sizing tools because size is the key input to most software models. Figure 1.1-1 depicts a multipurpose software cost data base.

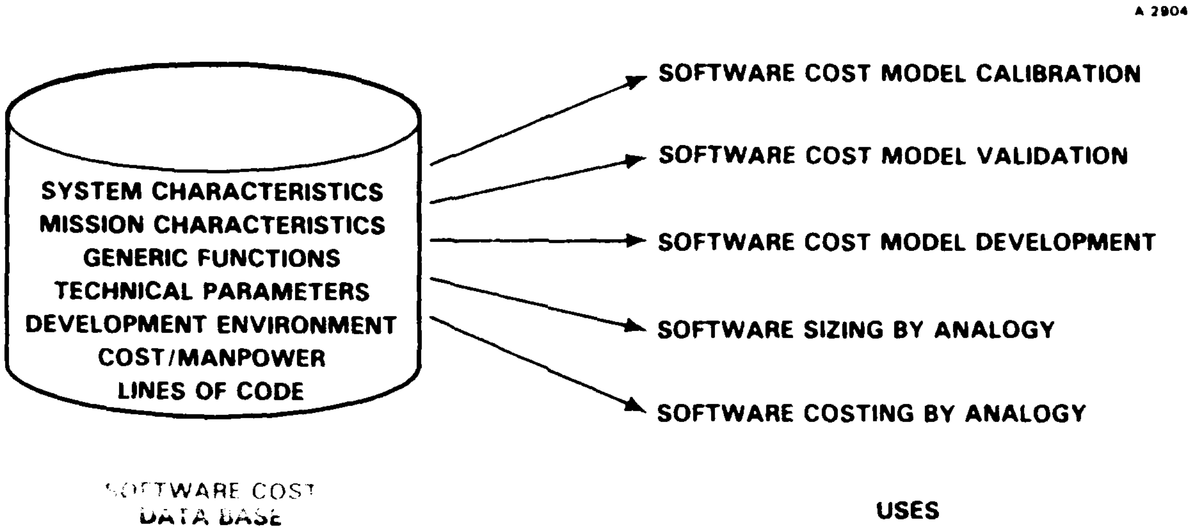


Figure 1.1-1 Software Cost Data Base Implementation

In addition to defining a software cost data base, this effort includes the design of data collection formats, the development and implementation of an approach for the initial data collection, and the establishment of a methodology for maintenance and growth of the data base in the future. Recommendations for future uses of the data base and approaches for advancing the state-of-the-art of software cost estimation are also part of this effort.

1.2 GENERAL APPROACH

The data base design is the result of discussions with government and industry personnel involved in the software development process as either software engineers, cost estimators or data base developers. It was further refined by using information from surveys of reports on previous data collection and data base development efforts, by review of the

user documentation for numerous software cost estimating models, and by analysis of prior software productivity, quality and reliability metrics research.

A standard software work breakdown structure (WBS) was defined and is used for the data base structure. The selected structure is based on an analysis of historical WBSs used at ESD, other structures proposed by industry and a comparison with WBS practices for hardware.

Based on the results of this research, data collection formats were then developed and distributed to potential participants. Analysis of the feedback from the participants and the completed data collection formats were used to refine the data collection package.

Finally, the software cost estimation requirements of ESD were analyzed to determine what enhancements were needed to improve the current capability.

1.3 REPORT ORGANIZATION

This report consists of two volumes: Volume I - a report on the data base designs and data collection methodology; Volume II - a compilation of the data collected. The following describes the contents of Volume I.

Chapter 2 of this report details the research performed to determine the data base contents and describes the elements that are included, as well as the structure of the data base. In Chapter 3 the development and implementation of the data collection formats and methodology are discussed and evaluated;

recommendations for growth of the data base are also made. Chapter 4 details future efforts that should be undertaken to continue improving the state-of-the-art of software cost estimation. A summary of the study results and recommendations is presented in Chapter 5. Appendix A contains cross reference tables between the cost model input parameters and data collection formats. The final version of the data collection package is contained in Appendix B.

2. DATA BASE REQUIREMENTS ANALYSIS

The software development process represents the complex interaction of requirements and resources to produce a software product. Figure 2-1 depicts some of the major categories of factors which affect the outcome of this process. The software data base must contain the data elements necessary to measure and evaluate the impact of these factors.

The contents and structure of the data base were determined through an analysis of the data reporting practices used and the data bases maintained by organizations within government and industry. Of particular interest were those software data bases whose objective was to support cost estimating or to

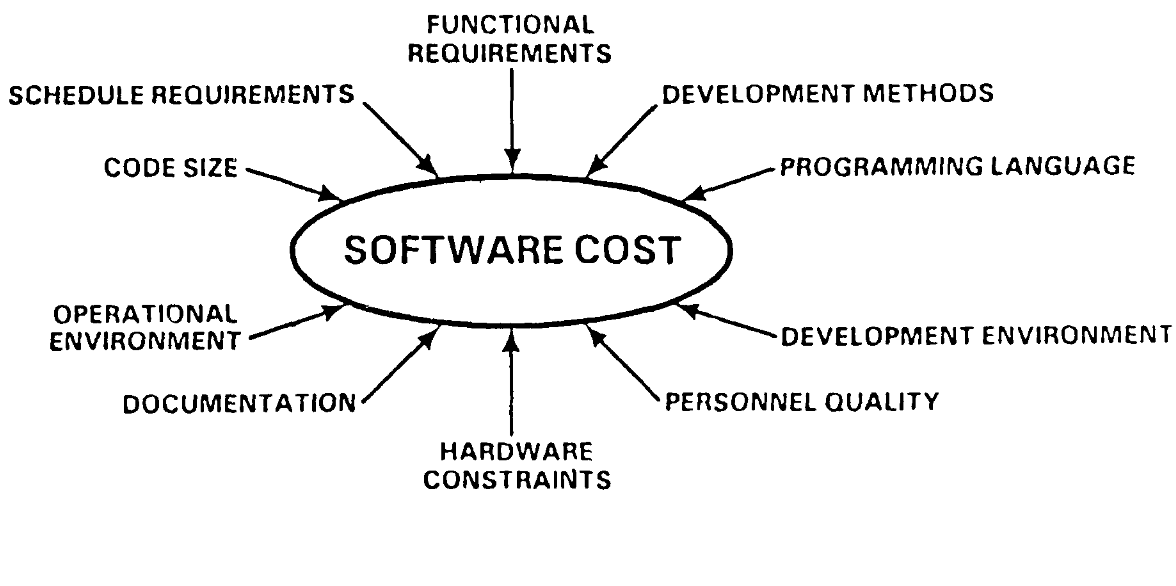


Figure 2-1 Factors Affecting Software Cost

evaluate software productivity. Likewise, the data requirements for use and calibration of the major software cost models in use today were reviewed. Finally, additional research was performed in the area of software productivity and quality metrics to evaluate alternatives to the current methods of software cost estimation.

2.1 EXISTING DATA BASES

Currently, software data bases fall into two categories: those that contain summary data at the system level and those that contain detail data at the lowest level to which software can be logically subdivided. To a great extent the amount of detail is determined not by the requirements of the data base developer, but by the availability of the data. The most detailed data bases exist within the development organizations which have a direct influence on the type of data collected for the day-to-day management of a development effort. Therefore, government software development organizations, such as the NASA Software Engineering Laboratory, and defense contractors historically have the best data available. On the other hand, data availability at government program offices is limited by the existing data items used for reporting software technical and resource utilization data.

Over the past few years the Data and Analysis Center for Software (DACS), the National Security Agency (NSA), and ESD, among others, have endeavored to develop new data items and reporting methods to obtain software data with sufficient detail to support the cost estimating process and to develop better estimating tools. The ESD efforts have resulted in the development of the Software Acquisition Resource Expenditure (SARE) Data Collection Methodology, which was completed in

December 1983 (Ref. 6). Both DACS and NSA are evaluating SARE to determine its applicability to their data needs.

In order to capitalize on the effort expended in the development of SARE, this project used SARE as the starting point for the data base requirements analysis. Additions have been made to the specific elements collected to provide information for definitive classification of the system and its Computer Program Configuration Items (CPCIs) and to provide cross checks for other data entries. On the other hand, some of the detail required by SARE has been eliminated because it is not practical to collect. For example, no data are collected at the Computer Program Component level with the exception of function and sizing data if that is the lowest level at which these data are available. Additionally, parameters which cannot be reasonably determined outside of the development organization at the time when an estimate is being prepared were also deleted.

2.2 SOFTWARE COST MODEL DATA REQUIREMENTS

A thorough review of the documentation for the major software cost models in use today (Refs. 2 through 5 and 7 through 14) resulted in the tabulation of the common elements among them. Appendix A contains cross reference tables showing the relationship between the data requirements for the COCOMO, JS-1, SLIM, and PRICE-S models and the data items on the data collection formats. For those items which were subjective in nature, the guidelines for making the value judgement were reviewed to identify any objective factors or characteristics that could be used to make the proper determination. This was not feasible in all cases, e.g., the complexity factor used by COCOMO has extensive guidelines which would require an inordinate number of numerical statistics to replace the subjective

determination which would be made by a software engineer familiar with the project. Whenever there was an overlap among the models, data representing the lowest common denominator and with the greatest level of granularity were included in the data base.

In areas where ambiguity may exist even with extensive data entry guidelines, data elements on which the ambiguous parameter would be based were also included in the data base to provide consistency checks. For example, an assessment of the CPU memory constraint as a percent utilization of the total available memory does not necessarily indicate a true memory constraint. Therefore, information about the expansion capability of the target computer and the reserve memory requirements is also included.

2.3 SOFTWARE PRODUCTIVITY AND QUALITY METRICS

The use of lines of code as a measurement of software productivity yields a wide range of results depending on the project. Although alternatives to lines of code have been proposed over the years, it appears that lines of code will remain the standard for cost estimation purposes. Many of the alternative approaches (Refs. 15 through 24) require the measurement of parameters that can only be determined after the completion of the software development effort, such as weighted statement count and process (a measure of the number of data items and paths in a program). Still others are highly correlated to lines of code, for example, number of modules. In general, the research performed on software productivity measurement indicates that, for cost estimation, lines of code is the most promising metric when used in combination with qualitative information.

Therefore, when lines of delivered source code are used as the primary metric, the variation in productivity from one project to another must be accounted for by qualifying the amount of code written using additional productivity metrics and by the addition of quality metrics. Many productivity metrics are nonquantitative, such as functions performed, development constraints, and personnel quality. These can be used strictly as qualitative metrics for categorization of development projects into homogeneous groupings. Alternatively, they can be quantified using a relative scale, as has been done in the COCOMO and JS-1 models, with detailed qualitative guidelines for selecting a numeric value. The same factors apply to quality metrics, such as reliability, maintainability, or completeness.

Both approaches are valid and should be used in conjunction with one another. To the extent that a sufficiently large data base exists, the available estimation models should be calibrated using the most narrowly defined grouping of projects possible. The development of new models should also be based on the most homogeneous grouping of projects that can be selected through the use of productivity and quality metrics and still contain sufficient data points to be statistically valid.

This data base is designed to include the metrics themselves, based on a definitive set of guidelines, or the lower level data elements required to derive the metrics.

2.4 DATA BASE CONTENTS

The data base can be logically divided into six distinct categories:

- System description and characteristics
- Development schedule data
- Hardware characteristics and constraints
- Development resources and constraints
- Software size and characteristics
- Resource expenditure data.

The data within each of these categories consist of the elements required to classify the system, to define the development environment, and to derive the software development cost drivers and input parameters for software cost and sizing models.

2.4.1 System Description and Characteristics

A key element of any data base design is the development of a classification scheme that can be used to group data for analysis and for data retrieval. The development of a comprehensive list of keywords based on the descriptions of a weapon system at the total system level is essential for matching a new system against those historical data which are most appropriate for model calibration. By extending this technique to the software segment of the system, to the CPCI level, and on down to the module level, a more restrictive selection can be made as the new system is defined in greater detail. Figure 2.4-1 shows the elements used for this progressive classification of a system.

Descriptive data defining the mission of the system, the hardware interfaces, the functions performed by the system as a whole and by the lower level components of the system are included in this data base so that keywords for data retrieval and classification can be developed. As the number of projects

SOFTWARE DEVELOPMENT PROJECT

- MISSION DESCRIPTION
- SYSTEM FUNCTIONS
- TARGET COMPUTER CHARACTERISTICS
- HARDWARE INTERFACES
- SOFTWARE FUNCTIONS

COMPUTER PROGRAM CONFIGURATION ITEM

- FUNCTIONAL DESCRIPTION
- SIZE PROFILE BY OPERATION
- SOURCE STATEMENT TYPE MIX
- PROGRAMMING LANGUAGE
- DEVELOPMENT PERSONNEL QUALITY
- OPERATIONAL RESPONSE REQUIREMENT
- DISPLAY REQUIREMENT
- TARGET COMPUTER CHARACTERISTICS

MODULE OR UNIT

- FUNCTIONAL CATEGORY
- PROGRAMMING LANGUAGE

Figure 2.4-1 Software Classification Hierarchy

documented in the data base increases, some of the other data categories, such as hardware constraints, can also be used to further restrict data selection or to provide a more homogeneous grouping.

2.4.2 Development Schedule Data

Schedule data are collected by major development milestone at the system level and for each CPCI in the system.

These milestones define the schedule to a level of detail sufficient to segment the effort into the major phases of the software life cycle, from project start to the start of the maintenance phase. These data, in conjunction with the monthly resource expenditure data, are used to determine the relative effort expended for each phase of the development. Both the original schedule dates, as well as the actual or estimated milestone completion dates, are included so that an assessment can be made of the degree of schedule acceleration. A subjective rating of the perceived schedule acceleration or stretchout is also included to determine its impact on the development.

2.4.3 Hardware Characteristics and Constraints

Information about the target computer is necessary to determine the timing and sizing constraints under which the software is being developed. Central processing unit (CPU) memory and timing utilization data are specified with supplemental information indicating the use of extraordinary measures to reduce size and the amount of software which is time constrained. Additional data about the expansion capability of the hardware, the reserve memory and time requirements are also included to determine whether the constraints indicated are real or perceived.

To allow for the grouping of software developments by class of target computer, especially for standard architectures, the computer used is specifically identified and the maturity of the hardware and the virtual machine is assessed. Instances of concurrent hardware and software development are indicated.

2.4.4 Development Resources and Constraints

There is a general consensus that the development environment and personnel factors are major contributors to the variation in productivity among software developments. This data base is designed to characterize the major components of the development environment which impact software cost.

The tools and methods available within the development organization are identified at the system level and their usage is rated for each CPCI. The characteristics of the development computer are compared with those of the target computer. Since access to the computer resource has a direct impact on productivity, the availability of the computer, as well as the access mode and computer location, are specified.

In the area of hardware cost estimation, it has been accepted practice to apply an improvement curve to the production of multiple units of an item. Although a direct analogy cannot be made for applying an improvement curve to software development, the experience of the development personnel in the weapon system application and with the development tools and techniques used will impact the overall productivity on a project. While software developers are involved in the familiarization process with the development environment, the design and code produced are often less than optimal and result in increased failure rates and redesign effort. Therefore, information about the experience of the personnel assigned at the start of the project is captured in the data base.

In addition to the effects of personnel quality on the costs of a project, the availability of personnel will likewise affect the schedule and manloading profiles for a project. Although these data are not a direct input into any

of the models, they can be used to analyze calibration results, in particular for SLIM which uses project duration, as well as development manpower, in the calibration process with the assumption that manloading is unconstrained.

2.4.5 Software Size and Characteristics

The type of software size data included in the data base is driven by two requirements:

- The need for size data at the CPCI level with allocations to various functional characteristics, processing modes, and languages to support the specific requirements of several cost models
- The need for size decomposition to the lowest level available with functional categorization and language identification to support sizing by analogy requirements.

The model-specific allocation data at the CPCI level, such as source statement mix, can also be used to group CPCIs into homogeneous groupings for model calibration and model development research. The lower level of functional detail provides data for in-depth analysis of variations in productivity or calibration results among a group of programs that appear to be homogeneous at the CPCI level.

To further clarify the magnitude of the development task, additional data about the amount of reuseable or modified code is required. This element will be particularly useful for evaluating the impact that Ada will have in this area.

2.4.6 Resource Expenditure Data

Ordinarily, cost data are obtained in terms of dollars. However, there are many problems inherent in that approach.

First of all, the data must be normalized to a common base year to eliminate the distortion which is caused by inflation and to make data from different time frames comparable. Next, the effects of different labor rates due to the geographic locations of the developers must be accounted for in the data. Finally, the data must be normalized for the impact of different overhead rates and charging practices for other direct costs. This normalization process is very intricate and would require the collection of additional data elements, such as labor and overhead rates. In many instances, these data are proprietary to a particular company and difficult to obtain because they are competitive sensitive.

Most data normalization problems are avoided when the cost data are collected in terms of manpower. The only adjustment required in this case is the conversion of the data to a common unit of manpower measurement, such as manmonths composed of a specific number of manhours. The labor content of the manpower is evaluated to insure that the data for all projects represent the same types of activities, for example, designers and programmers, but not clerical support.

The ESD data base design requires that resource utilization data be expressed in equivalent manmonths (176 hours) and timephased in terms of months after contract award or project start. The data are timephased so that they can be used to evaluate the Rayleigh curve timephasing used by SLIM and JS-1 and the resource allocation by phase method in COCOMO. They can also be used to select an appropriate method from the choices offered in PRICE-S.

2.5 WORK BREAKDOWN STRUCTURE

Since the work breakdown structure (WBS) is the scheme generally used for organizing cost data and for reporting cost performance on DoD programs, the WBS methodology has been selected as the structure for organizing the software cost data base. This decision required the development of a standard software work breakdown structure for integration into a project WBS defined in accordance with MIL-STD-881A (Ref. 25). In order to facilitate comparison of projects at the lower levels of detail, ESD must adopt a standard WBS with detailed definitions comparable to those in MIL-STD-881A. Ideally, this standard would be accepted by the other DoD software developers.

The WBS definition began with a review of the WBSs in use for cost performance data reporting at ESD. The analysis encompassed all active ESD programs, as well as projects completed within the last ten years. In general, older projects did not use a detailed WBS for software and in many cases the software effort was located below the reporting level. Over the years, the level of software detail reported has been increasing in parallel with the growth in the importance of software for ESD systems. The majority of projects now obtain cost data at the CPCI level.

The results of the above review were then compared with several proposed software work breakdown structures (Refs. 6, 26, 27, 28). In contrast to the product-oriented structure identified in MIL-STD-881A, many of the proposed approaches have either an accounting or a functional orientation. However, these other approaches are not incompatible with a product-oriented WBS, since the functional and accounting shredouts can be made within a product-oriented WBS below the lowest product level of interest. For software developments the CPCI

level defines a natural subdivision of the product and is the level most commonly used for cost estimating. Breakouts below this level are either artificial, such as the CPC, or are at too great a level of detail for cost effective data collection, such as the module. Therefore, the CPCI has been selected as the key element to be included in the software work breakdown structure. Extension of the WBS below this level will be left to the discretion of the developer.

Above the CPCI level, a software effort can be logically subdivided at the system level into one or more subsystems. Each subsystem can then be divided into one or more CPCIs. Although a software subsystem is a logical and not a physical entity, it is still analogous to a hardware subsystem and should be treated in the same manner. The recommended WBS shown in Table 2.5-1 is an enhancement of a MIL-STD-881A WBS with software subsystems in parallel with hardware subsystems at level three. Support software is treated as a separate software subsystem. Each software subsystem is extended to level four into a breakout of the subsystem design activity, the subsystem integration, and the CPCIs of which it is composed. This WBS is based on an alternative presented in SARE. The other alternatives in SARE were rejected because they do not cover any situations that cannot be handled within the recommended approach and could result in confusion and inconsistent application.

During the data collection effort the recommended WBS was discussed with several defense contractors and the general reaction was favorable. The recommended WBS was similar to those being adopted by these contractors for their internal management requirements. These contractors also stressed the need for standardization.

TABLE 2.5-1

DEFENSE SYSTEM WORK BREAKDOWN STRUCTURE

Level 1	Level 2	Level 3	Level 4
Defense System			
	Prime Mission Equipment		
		Integration and Assembly	
		Hardware Subsystem or End Item 1	
		.	
		.	
		.	
		Hardware Subsystem or End Item n	
		*Software Subsystem 1	
			Subsystem Analysis & Design
			Subsystem Integration & Test
			Computer Program Configuration Item 1
			.
			.
			.
			Computer Program Configuration Item n
		.	
		.	
		.	
		*Software Subsystem n	
		*Support Software	
			Computer Program Configuration Item 1
			.
			.
			.
			Computer Program Configuration Item n
	Training		
		Equipment	
		Services	
		Facilities	
	Peculiar Support Equipment		
		Organizational	
		Intermediate	
		Depot	
	System Test & Evaluation		
		Development Test & Evaluation	
		Operational Test & Evaluation	
		Mockups	
		Test & Evaluation Support	
		Test Facilities	
	System/Program Management		
		Systems Engineering	
		Project Management	

*Replaces Computer Program at Level 3 from MIL-STD-881A

TABLE 2.5-1

DEFENSE SYSTEM WORK BREAKDOWN STRUCTURE (Continued)

Level 1	Level 2	Level 3	Level 4
	Data		
		Technical Publications	
		Engineering Data	
		Management Data	
		Support Data	
		Data Depository	
	Operational/Site Activation		
		Contractor Technical Support	
		Site	
		Construction	
		Site/Ship/Vehicle Conversion	
		System Assembly Installation & Checkout on Site	
	Common Support Equipment		
		Organizational	
		Intermediate	
		Depot	
	Industrial Facilities		
		Construction/Conversion/Expansion	
		Equipment Acquisition or Modernization	
		Maintenance	
	Initial Spares & Initial Repair Parts		

2.6 DATA BASE STRUCTURE

The data base is organized using the recommended WBS. At level one, the defense system, data describing the system mission, major functions, hardware interfaces, development tools and methodologies, system level documentation page counts, and change history are collected. Additional data describing product characteristics, the development environment, and development resources are collected at the CPCI level for those elements that are different for each CPCI. Schedule and failure data are also collected at this level.

Sizing and functional data are collected, at a minimum, to the CPCI level for cost estimation and down to the lowest level available for size estimation. Resource expenditure

data are collected for every element of the WBS that applies to software only. For projects that are entirely software, cost data are collected for all elements of the WBS.

During the data retrieval process, analogies are progressively developed starting at the defense system level to select homogeneous projects for further analysis, continuing at the CPCI level to select CPCIs for software cost model validation/calibration or development, and at the module level for software sizing. Depending on the size of the data base and the sample size requirements for a statistically valid analysis, restrictive criteria are selected from the characteristics and functions at the appropriate level to select a homogeneous subset of the data base.

3. DATA COLLECTION METHODOLOGY

The establishment of a data base from the design described in Chapter 2 required the development of data collection formats and a data collection methodology. The approach taken also provides for use of the formats for future data collection to maintain the data base and for data collection for cost estimating model input. Appendix A contains tables which cross-reference the COCOMO, SLIM, PRICE-S, and JS-1 cost model input parameters to the data collection formats. In addition to detailing the methodology developed for the initial collection, this chapter also proposes approaches for data base maintenance and growth.

3.1 COLLECTION FORMATS

The data collection formats were developed using the Mitre SARE (Ref. 6) formats as a strawman. Formats developed by DACS, NSA, NASA-SEL, and the Aerospace Corporation were also evaluated and the best elements were selected from each of the different approaches. Other elements represent new designs to enhance clarity and useablity or to add unique data items. Four separate formats were originally developed to allow for the different software decomposition levels at which data items would be collected. As a result of this initial data collection effort, a fifth format was developed to separately collect hardware data. The data collection package also includes a comprehensive set of instructions. Figure 3.1-1 summarizes the contents of each element of the data collection package.

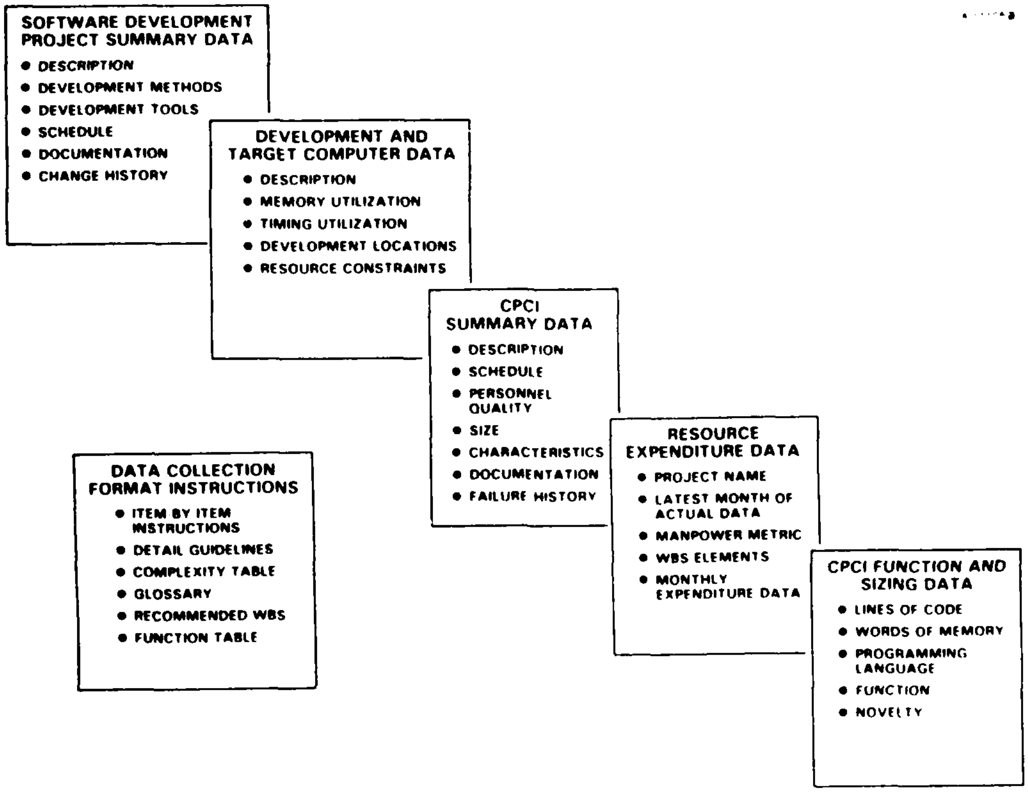


Figure 3.1-1 Data Collection Package

3.1.1 Software Development Project Summary Data

This format (Figure 3.1-2) is designed to collect data at the top level of the WBS, the defense system. It includes descriptions of the mission of the system as a whole, specification of the hardware interfaces required, identification of the system functions and the allocation of those functions to the software elements of the system. These data are used for classification of the system so that it can be grouped with similar systems for analysis and for model validation or calibration.

The format also contains a checklist of commonly used software development tools and techniques. These checklists are used in conjunction with data elements on the CPCI Summary Data format to develop inputs for several cost models requiring an assessment of the development environment. A schedule at

SOFTWARE DEVELOPMENT PROJECT SUMMARY DATA

1 Project Name ______________________ Date ________

2 Development Contractor/Organization ______________________

3 Project Description

3.1 Mission Description ______________________

3.2 Major Hardware Interfaces ______________________

3.3 Major System Functions ______________________

3.4 Major Software Functions ______________________

3.5 Number of CPCIs ________

3.6 CPCI Names ______________________

3.7 System User

Development Contractor ___ Other Commercial Company ___

Department of Defense ___ Other Government Agency ___

4 *Development Methodologies Used*

4.1 Specification

Functional ___ *Procedural* ___ *English* ___ Other ________

4.2 Design

Top Down ___ Bottom Up ___ Iterative Enhancement ___

Hardest First ___ None ___ Other ________

4.3 Development

Top Down ___ Bottom Up ___ Iterative Enhancement ___

Hardest First ___ None ___ Other ________

4.4 Coding

Simulating Construct ___ Structured Code ___ None ___

Other ________

4.5 Testing

Top Down(stubs) ___ Bottom Up(drivers) ___

Specification Driven ___ Structure Driven ___

None ___ Other ________

4.6 Validation/Verification(Inspection)

Peer Review ___ Walk Throughs ___ *Proof* ___

None ___ Other ________

4.7 Formalisms

Program Design Language(Specify) ________

HIPO Charts ___ Flowcharts ___ Chapin Charts ___ HOS ___

Other ________

5 Software Development Tools Used

Assembler ___	Basic Linker ___
Basic Monitor ___	Batch Debug Aids ___
Higher Order Language Compiler ___	Macro Assembler ___
	Simple Overlay Linker ___
Basic Source Editor ___	Language Independent Monitor ___
Basic Library Aids ___	Basic Data Base Aids ___
Real-time or Timesharing Operating System ___	Extended Overlay Linker ___
	Database Management System ___
Interactive Debug Aids ___	Simple Programming Support Library ___
Interactive Source Editor ___	Virtual Memory Operating System ___
Database Design Aid ___	Simple Program Design Language ___
Performance Measurement And Analysis Aids ___	Programming Support Library with Basic Configuration Management Aids ___
Set-Use Static Analyzer ___	Control Flow Static Analyzer ___
Basic Text Editor & Manager ___	Program Flow and Test Case Analyzer ___
File Manager ___	Full Programming Support Library ___
Documentation System ___	Project Control System ___
Requirements Specification Language and Analyzer ___	Extended Design Tools ___
	Automated Verification System ___
Fault Report System ___	Crosscompilers ___
Instruction Set Simulators ___	Display Formatters ___
Data Entry Control Tools ___	Communications Processing Tools ___
Conversion Aids ___	Structured Language Tool ___
Other ________	Other ________
Other ________	Other ________

6 Development Schedule

Project Milestone	Original Date	Actual Date	Estimated Date
Contract Award			
System Requirements Review			
System Design Review			
Preliminary Design Review			
Critical Design Review			
Preliminary Qualification Test			
Formal Qualification Test			
Start CPCI Integration Into System			
Complete CPCI Integration into System			
Start Development Test & Evaluation			
Complete Development Test & Evaluation			
Start Initial Operational Test & Eval			
Complete Initial Operational Test & Eval			
Functional Configuration Audit			
Physical Configuration Audit			
Formal Qualification Review			
System Delivery			

7 Documentation

Document Title	# of Pages	Est. or Act.
System Engineering Management Plan		
Computer Program Development Plan		
System Test Plan		
Other ________		
Other ________		
Other ________		
Other ________		

8 *Software Change History*

Development Phase	# of Changes Approved	Est'd DSLOC Change (+/-)	Est'd Manpower Change (+/-)
Preliminary Design (Contract Award to PDR)			
Detailed Design (PDR to CDR)			
Code & Debug (CDR to Test & Integ Start)			
Test & Integration (Test & Integ Start to FQT)			
System Test/IOC (FQT to Contract End)			

Figure 3.1-2 Software Development Project Summary Form

the total project level is included to facilitate allocation of resource expenditure data to the different phases of the software development process.

System level documentation page counts are requested as a prospective variable for use in estimating documentation cost and for measuring software maintainability. A software change history is also included for an evaluation of software requirements volatility and its impact on software cost.

A copy of this format is prepared for each project and is used both for additions to the data base and to collect cost model input parameters for an estimate. It can also be used to collect data for analogy selection at the project level.

3.1.2 Computer Hardware Detail Data

This format (Figure 3.1-3) is designed to collect data about the target computer on which the software product will operate and about the computer on which the software is being developed. The constraints of the target computer are identified to define the operational environment for the software. The memory and time utilization by the software is specified, along with reserve requirements and expansion capability, so that an assessment can be made of constraints on the development process caused by these operational requirements. The development computer characteristics and its effectiveness as a resource are likewise addressed.

This format is prepared for each target computer in the system and its related development computer. It is used for additions to the data base and for collection of cost model input parameters.

DEVELOPMENT AND TARGET COMPUTER DATA FORM

1. Target Computer
   - 1.1 Manufacturer ____ Model Number ____
   - 1.2 Main Memory Size in Words ____ Word Size ____ Bits
   - 1.3 Maximum Main Memory Size ____
   - 1.4 CPU Processing Speed ____
   - 1.5 Reserve Memory Requirement ____%
   - 1.6 Reserve Timing Requirement ____%
   - 1.7 Concurrent Development with Software Yes __ No __
   - 1.8 Virtual Machine Volatility
     Very Low __ Low __ Nominal __ High __ Very High __
   - 1.9 Percent Utilization

	≤50%	51-70%	71-85%	86-95%	>95%
A. CPU Memory	__	__	__	__	__
B. Execution Time	__	__	__	__	__

   - 1.10 CPU Memory Constraint Evaluation
     - No Memory Economy Measures Required __
     - Some Overlay Use Or Segmentation Required __
     - Extensive Overlay And/Or Segmentation Required __
     - Complex Memory Management Economy Measures Required
   - CPU Time Constraint Evaluation
     - No Software is CPU Time Constrained __
     - ≤10% of Source Code is Time Constrained __
     - 50% of Source Code is Time Constrained __
     - 75% of Source Code is Time Constrained __
   - 1.11 CPCIs Hosted on This Computer ____

2. Development Computer
   - 2.1 Same as Target Computer Yes __ No __
     If No, Manufacturer ____ Model Number ____
     Difference Between Development and Target Computer ____
   - 2.2 Turnaround Time
     Low (interactive, specify) ____ Nominal (<4hrs) __
     High (4-12hrs) ____ Very High (>12hrs) ____
   - 2.3 Percentage of Source Instructions Developed Using Each of the Following Access Modes (Total=100%)
     - A. Batch ____%
     - B. Dedicated Processor ____%
     - C. Test Bed with High Priority ____%
     - D. Test Bed with Low Priority ____%
     - E. Interactive ____%
     - F. Other ____ ____%
   - 2.4 For Interactive Development, Indicate the Average Number of Software Engineers/Programmers per Terminal ____
   - 2.5 Number of Development Sites ____
   - 2.6 Development Site Locations ____
   - 2.7 Development Computer Location(s) ____
   - 2.8 Software Development Tool Usage
     Very Low __ Low __ Nominal __ High __ Very High
   - 2.9 Development Computer Resource Availability ____

Figure 3.1-3 Development and Target Computer Data Form

3.1.3 CPCI Summary Data

This format (Figure 3.1-4) is designed to collect data at the CPCI level. The data required at this level are determined primarily by the input parameters of the COCOMO, SLIM, PRICE-S, and JS-1 cost models.

The format begins with the name of the CPCI and a narrative description which can be used for analogy selection and homogeneous grouping of CPCIs. Next, the experience of the development personnel relative to the development environment and the software application, as well as the quality of the personnel, is assessed.

The balance of the format is concerned with the characteristics and size of the software. As required by several of

COMPUTER PROGRAM CONFIGURATION ITEM SUMMARY DATA

1. CPCI Name ________

2. Functional Description ________________

____

____

3. Schedule

3.1 Milestone Data

CPCI Development Milestones	Original Date	Actual Date	Estimated Date
Design Start	____	____	____
Preliminary Design Review	____	____	____
Development Specification Approval	____	____	____
Critical Design Review	____	____	____
Start Coding/Debugging	____	____	____
Complete Coding/Debugging	____	____	____
Start Informal Integration & Test	____	____	____
End Informal Integration & Test	____	____	____
Preliminary Qualification Test	____	____	____
Formal Qualification Test	____	____	____
Product Specification Approval	____	____	____
Functional Configuration Audit	____	____	____
Physical Configuration Audit	____	____	

3.2 Schedule Acceleration/Stretchout Assessment

Below 75%   75-85%   86-100%   [illegible]   [illegible]

4. Personnel

4.1 Average Experience

	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
A. Application Area						
B. Techniques Used						
Languages Used						
Virtual Machine						
E. Support Software Tools						

4.2 Average Personnel Quality Percentile

0-15% __   16-35% __   36-55% __

56-75% __   76-90% __   90-100% __

4.3 Manpower Availability ____ %

4.4 Peak Manloading ____

5. Reliability Requirement

Very Low __ Low __ Nominal __ High __ Very High __

6. Complexity

Very Low __   Low __   Nominal __

High __   Very High __   Extra High __

7. Quality of Specification

Very Precise __   Precise __   Imprecise __

8. Size

8.1 Deliverable Lines of Source Code Excluding Documentation ____

8.2 Lines of Source Code Documentation ____

8.3 Data Base Size in Bytes or Characters ____

8.4 Size Breakdown by Operation As a Percent of Item 8.1 (Total = 100%)

A. Data Storage & Retrieval ____ %

B. Online Communications ____ %

Real-Time Command & Control ____ %

C. Interactive Operations ____ %

E. Mathematical Operations ____ %

F. String Manipulation ____ %

G. Operating Systems ____ %

H. Other ________ ____ %

I. Other ________ ____ %

8.5 Operational Response Requirements Distribution As a Percent of Item 8.1 (Total = 100%)

A. Real-Time ____ %   B. On-Line ____ %

Time-Constrained ____ %   D. Non-Time Critical ____ %

8.6 Source Statement Type Mix As a Percent of Item 8.1 (Total = 100%)

A. Logical ____ %   B. Command ____ %   Mathematical ____ %

D. Data Manipulation ____ %   E. Data Declaration ____ %

8.7 Special Display Requirements

Simple Input/Output __   User Friendly __

Interactive __   Complex Requirements Severe Impact

8.8 Languages Used as a Percent of Item 8.1 (Total = 100%)

A. Language ________ Percentage ____ %

B. Language ________ Percentage ____ %

C. Language ________ Percentage ____ %

D. Language ________ Percentage ____ %

8.9 Reusable Code From Similar Projects

Project	# of DSLOC	% of Modification Design	Code	Required Integration
____	____	____ %	____ %	%
____	____	____ %	____ %	%
____	____	____ %	____ %	____ %
____	____	____ %	____ %	____ %

9. Documentation

Document Title	# of Pages	Est.	Act.
CPCI Development Specification			
CPCI Product Specification			
Test Plan			
Test Procedures			
Test Report			
User's Manual			
Operator's Manual			
Other ____			
Other ____			
Other ____			

10. Software Failure History

Development Phase	# of Software Failures/Errors
Preliminary Design (Contract Award to PDR)	____
Detailed Design (PDR to CDR)	____
Code & Debug (CDR to Test & Integ Start)	____
Test & Integration (T & I Start to FQT)	____
System Test (OK FQT to Contract End)	

Other Factors or Characteristics ________________

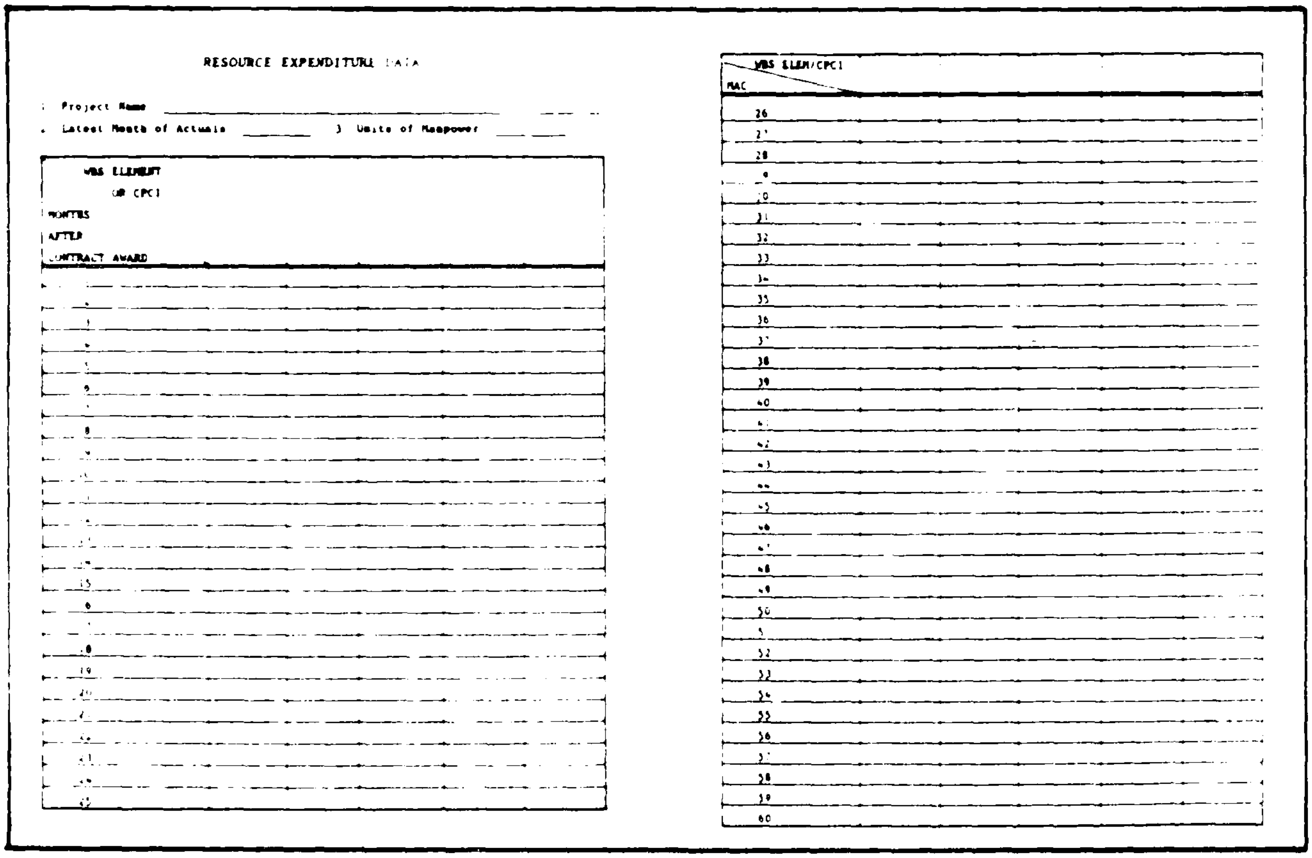
Figure 3.1-4 Computer Program Configuration Item Summary Data Form

the cost models, the software size is allocated to different functional operations, source statement mixes, programming languages, and operational modes. These size profiles can also be used for developing analogies. Several software quality measures required by the models are included. Additionally, two prospective measures of software quality, that is, documentation page count and failure history, are collected for future research.

This format is prepared for each CPCI in the system and can be used for additions to the data base or to collect cost model input parameters.

3.1.4 Resource Expenditure Data

This format (Figure 3.1-5) is designed to collect data on manpower resource expenditures for a project by CPCI, WBS


RESOURCE EXPENDITURE DATA

1. Project Name

2. Latest Month of Actuals

3. Units of Manpower

WBS ELEMENT OR CPCI

MONTHS AFTER CONTRACT AWARD

WBS ELEM/CPCI

MAC

Figure 3.1-5 Resource Experditure Data Form

element or other aggregation of data. For this initial data collection effort, it allows the data provider to supply information to the lowest level of detail that is available. Aggregation of the data to appropriate levels for inclusion in the data base was performed after the receipt of the formats. If this format is used for collection of data on future efforts, the WBS items to be included can be specifically identified as a requirement.

The format is structured to collect data by month relative to the contract award or project start date. For ongoing projects, the last month for which actual data are available is indicated; estimated data are used for the balance of the development project. In order to minimize the data conversion required on the part of the format preparer, manpower data can be included in any units as long as the basis for the units is identified, for example, manmonths representing 152 manhours of effort.

This format is used for adding projects to the data base and up to five WBS elements with 60 months of data can be entered on each form. It does not contain any entries required for cost model inputs; however, it will likely be useful for updating cost estimates for ongoing projects.

3.1.5 CPCI Function and Sizing Data Detail

This format (Figure 3.1-6) is designed to collect software size and functional data to the lowest level of detail available. It constitutes a decomposition of a CPCI down to the module level. The function definition is selected from a table of electronics systems software functions contained in the instructions. Since this table is not all-inclusive, the format preparer can specify a unique function category for an element.

CPCI NAME: ______

CPC NAME	MODULE OR UNIT NAME	REUSED CODE LITTLE OR NO MODIFICATION	REUSED CODE EXTENSIVE MODIFICATION	NEW CODE	FUNCTION	SIZING		PROGRAMMING LANGUAGE
						# OF DISLOC	# OF WORDS	

Figure 3.1-6 Computer Program Configuration Item Function and Sizing Data Detail Form

Each module is characterized according to its degree of novelty to clarify the magnitude of the development task:

- Reused Code - little or no modification
- Reused Code - extensive modification
- New Code.

To increase the flexibility of the data base, sizing data are collected in two ways:

- Lines of source code, excluding comments
- Words of CPU memory occupied.

Size information is requested in lines of source code because that is the measure currently used by most models. Words of memory occupied are also requested so that object instructions can be estimated, since most sizing analyses performed during system development are done in memory words. Moreover, PRICE-S requires size data as executable machine instructions, which can be more accurately derived from the memory word count than from lines of source code. The language in which each subelement is programmed is also identified on this format.

This format is prepared for each CPCI and documents each subelement into which the CPCI is decomposed. The primary purpose of these data are for use in software sizing models. They can also be used to cross-check the size distributions on the CPCI Summary Format and to evaluate the memory utilization efficiency of different languages.

3.1.6 Data Collection Format Instructions

The data collection format instructions begin with a description of the objective of the data collection effort and the five data collection formats. A separate section of instructions for each format follows with item-by-item instructions. Definitions of specific data items are part of the item instruction, if appropriate. Subjective items include a detailed set of guidelines drawn from cost model instructions to facilitate consistent evaluations across a range of projects.

A glossary defines any software engineering terms used in the instructions, as well as software development milestones and reviews. This glossary was developed using the draft MIL-STD-SDS (Ref. 29) as the primary source of definitions; this glossary should be revised to reflect the final version of the standard when it is issued. Additional items were taken from

Refs. 25 and 30 through 33. Finally, Refs. 2, 3, 4 and 5 were used for model specific definitions.

The instruction package also contains a table of standard software function categories, developed under the SARE effort, to be used in completing the CPCI Function and Sizing Detail Format and the recommended work breakdown structure to be used for the Resource Expenditure Form.

Although the instruction package is sizable, it is the minimum required to insure consistent interpretation of the data requirements. It eliminates much of the additional explanation that is given verbally when detailed instructions are not available in writing. The initial feedback from industry has been that the package is quite clear and well-conceived. Residual questions from participants in the data collection have been relatively few.

3.2 COLLECTION APPROACHES

Three different approaches were selected for the initial data collection:

- Method 1 -- Data collection forms completed by defense contractor
- Method 2 -- Data forms completed by program office
- Method 3 -- Data forms completed by TASC analyst using program office documentation.

The relative efficiency of these approaches was evaluated when the data collection formats were completed.

Method 1 depends on the willingness of companies to divulge proprietary data about Government software projects to a third party. On this project, cooperation of the defense industry was obtained through the use of previously established professional contacts. Because of TASC's corporate policy of not contracting with defense contractors, the participants had no concerns about a possible conflict of interest. Typically, after the initial contact was established and a commitment to furnish data secured, constant follow-up activity was required to ensure success. This follow-up activity included a visit to the contractor facility to gather feedback about the data collection formats, to identify areas of ambiguity, and to evaluate the quality of the data furnished. Each participant was offered an aggregated, non-attributable tabulation of the data furnished by all of the participants for use in evaluating his own data.

Method 2 is dependent on establishing contact in the program office with an appropriate individual familiar with the software effort for the project. The reward offered for participation, that is, improved software cost estimating support in the future, is less tangible for this approach. Therefore, the follow-up activity must be even greater than with the first approach. Since the data sources at the program office are limited primarily to the existing data reporting items, the program office contact must interact extensively with the development contractor, especially for personnel characteristic and manpower utilization data.

Method 3 is dependent on the quality and completeness of documentation available for a project. Since the analyst performing the data extraction is not usually familiar with the program, the process can be very time consuming and not all of the data elements required are available in the documentation. Therefore, additional effort was required to obtain missing data from the program office or the software developer.

All three approaches were used on this effort, although not with the same emphasis. The majority of the data was obtained through the defense contractors, because they have more direct access to the data, minimizing the potential for errors and misinterpretation. Table 3.2-1 summarizes each approach with its prerequisites, advantages and disadvantages based on the results of this effort.

3.3 DATA BASE MAINTENANCE AND GROWTH

The data collected under this effort is documented in Volume II of this report. Although it provides a solid basis for initial analysis and model calibration, the continued evolution of the software development process and the need for additional data points to permit more narrow definition of homogeneous groups requires a dynamic data base. Opportunities currently exist for immediate additions to the data base; however, they must be supplemented with a mechanism for regular data collection during the system acquisition process.

3.3.1 Data Collection Methodology

There are four methods that can be employed for adding projects to the data base created under this initial effort:

- Use the final version of the data collection package developed under this effort "as is" (Appendix B)
- Create a formal Data Item Description (DID) from the collection package developed under this effort
- Formally modify existing or planned data item descriptions to include the data elements specified in the data base design

TABLE 3.2-1

DATA COLLECTION APPROACH COMPARISON

APPROACH	PREREQUISITES	ADVANTAGES	DISADVANTAGES
Method 1 - Development Contractor as Primary Source	Established personal contacts in industry Guaranteed anonymity to data source Independent data collector Benefit to data source	Better quality and more detailed data Contractor charges cost to overhead	Difficult to obtain timely response Frequent follow-up by data collector Practical for completed projects only No calibration for specific contractors
Method 2 - Government Program Office as Primary Source	Established working relationship with program office Contractual relationship between program office and development contractor Program office personnel with experience on project Benefit to data source	Practical for on-going or recently completed programs Allows model calibration for a specific contractor	Not effective with non-ESD program offices Consumes program office resources Development contractor participation charged to contract Some follow-up by data collector
Method 3 - Deliverable Documentation as Primary Source	High quality and complete documentation	Allows model calibration for a specific contractor Uses only data collector resources	Current documentation practices inconsistently applied Additional data required from program office and development contractor Interpretation of data by inexperienced personnel Highly time-consuming for data collector

- Prepare instructions to be used for tailoring existing data items to report data specified in the data base design.

The first method is appropriate for collecting data to derive the input parameters for the cost models used to develop an estimate or to support a source selection and to add completed projects to the data base. This collection package serves as a replacement for ASD Form 169a, since it includes not only the data collected on that form, but also additional data for cost models which are not covered by Form 169a. The last three methods are applicable for collecting data on new projects for the data base or for monitoring performance on those projects.

Implementation of the first two methods is based on the final version of the formats resulting from the feedback from the initial collection effort. If the formats are not going to be used as formal contract data requirement list (CDRL) items, the first method is used. Local form numbers are assigned and the package is ready for use. On the other hand, if the formats are going to be used as contractual deliverables, then they must be formally established as data items within the required review and approval cycle.

The last two methods are implemented by identifying the specific data items that are appropriate for reporting subsets of the data elements contained in the data collection formats. In Ref. 35, the preliminary report for this effort, the following list of data items with recommended changes was presented:

- Cost Performance Reports (DI-F-6000C and DI-F-6010C) - Modify/tailor format 1 of the Cost Performance Report and the Cost Schedule Status Report to show manpower data for the software WBS elements or

format 4 of the Cost Performance Report to show manpower by WBS rather than functional area

- Program Master Schedule (DI-A-3007 and DI-A-3009) - Identify specific software development milestones at the system and CPCI levels to be included in the schedule submissions

- System Specifications (DI-E-3101A, DI-E-3102A and DI-E-3117) - Add a format summarizing mission, functional, hardware component, and software component data

- Software Development Specification (DI-E-3119A) - Add a format summarizing the major software functions and the CPCIs

- Software Product Specification (DI-E-3120A) - Require functional categorization by standard category as part of the module descriptions

- Software Sizing and Timing Analysis Report (DI-S-3581) - Require sizing information at the module level in source lines of code, as well as CPU memory words; incorporate a format identifying the hardware configuration baseline against which the analysis is made and include information about expansion capability and reserve requirements

- Software Development Plan (R-DID-103) - Require a summary format identifying tools and techniques to be used, development computer characteristics and constraints, and an experience and quality profile of personnel assigned to the project.

If these changes are incorporated, most of the data elements required for the data base will be available directly. The balance can be derived through analysis and aggregation of detail data from the reports.

Formal modification of the existing data items will require the full review and approval cycle, while instructions for tailoring the data items can be implemented locally. However, since the Joint Logistics Commanders (JLC) Joint Policy Coordinating Group on Computer Resource Management is in the process of developing revised data item descriptions (Ref. 34) for reporting on software projects, formal modification of existing data items which are to be replaced by the JLC versions is not necessary. Instead, ESD should work to have their requirements incorporated into the JLC versions. Separate action will only be required to modify the cost performance data items. Table 3.3-1 summarizes the recommended changes to the JLC data item descriptions required to maintain the ESD Software Data Base.

The Program Master Schedule data item does not need modification; however, the appropriate milestones and levels of detail required for software must be specified in the contract. The changes to the other data items are required in order to insure that the data are provided consistently from one project to another and can easily be extracted for incorporation into the data base. The summary formats, which are to be added, should be standard formats with detailed instructions equivalent to those prepared under this effort. These formats should be augmented with standard tables of keywords for classifying systems, such as Table B-2 in Appendix B to this report. Table B-2 itself needs to be further refined to include other types of systems.

3.3.2 Additional Data Sources

Since this effort consisted of only an initial collection to develop a basic capability and field test the collection approach, several potential data sources which were identified

TABLE 3.3-1

SOFTWARE DATA ITEMS

DATA ITEM NUMBER	TITLE	CURRENT SOFTWARE DATA CONTENTS	RECOMMENDED CHANGES
DI-S-X101	System/Segment Specification	Major system functions Hardware/software inter-relationships Interface requirements (Internal/External) Performance requirements Programming language Compiler/assembler	Require a summary format with: Software Development Project Summary Data Form, Items 1, 2, 3 Development and Target Computer Data Form, Item 1 CPCI Summary Data Form, Items 5, 6, 7
DI-A-X103	Software Development Plan	Development resource requirements Development personnel requirements Tools and techniques Design coding and testing methodology Reusable off-the-shelf code	Require a summary format with: Software Development Project Summary Data Form, Items 4,5 Development and Target Computer Data Form, Item 2 CPCI Summary Data Form, Items 4, 8, 9
DI-E-X106	Software Problem/ Change Report	Problem description Cost/schedule impact Component/document affected Origination date	Provide for periodic summarization of individual problem reports by development phase
DI-E-X107	Software Requirements Specification	CPCI function Programming requirements Sizing and timing requirements Detailed functional requirements Data base requirements	Require a summary format with: Development and Target Computer Data Form, Items 1.9, 1.10, 1.11 CPCI Summary Form, Items 2,8 CPCI Function and Sizing Detail Form, all items

TABLE 3.3-1

SOFTWARE DATA ITEMS (Continued)

DATA ITEM NUMBER	TITLE	CURRENT SOFTWARE DATA CONTENTS	RECOMMENDED CHANGES
DI-E-X108	Interface Requirements Specification	Interface block diagram Software to software interface Hardware to software interface	None Required
DI-E-X109	Software Standards and Procedures Manual	Software development tools Software development methodology	Require a summary format with: Software Development Project Summary Data Form Items 4, 5
DI-E-X110 & DI-E-X114	Software Top Level Design Document Software Product Specification	Functional allocation to unit/module Sizing and timing budget allocation	Require a summary format with: CPCI Summary Data Form, Item 8 Development and Target Computer Data Form, Items 1.9, 1.10, 1.11 CPCI Function and Sizing Detail Form, all items
DI-F-6000C & DI-F-6010	Cost Performance Report Cost/Schedule Status Report	Monthly cost by WBS element Manpower by functional category	Modify cost format to include manpower by WBS
DI-A-3007 & DI-A-3009	Program Master Schedule	Original start & complete milestones Actual start & complete milestones	None Required

during the study were not actively pursued. These sources can be used to substantially increase the size of the data base in the near term.

Although ESD program offices were used as data sources on a limited basis, the bulk of the data was collected from defense contractors in order to concentrate on sources to which ESD does not have easy access. Therefore, ESD cost analysis personnel can collect data directly from those program offices which have ongoing or recently completed programs and were not used as sources during this effort, for example, SPADOC, WIS, and Berlin Radar.

In October 1983, The Aerospace Corporation completed a software sizing survey (Ref. 28) for the Air Force Space Division (SD) which contains size and function data for ten systems. In addition to the sizing data, Aerospace collected many of the same data elements that are included on the Project Summary, CPCI Summary, and Resource Expenditure Forms developed under this effort. Although many of the systems in the SD data base are space-borne, there are similarities in complexity, reliability requirements, documentation, and timing and sizing constraints, as well as the development environment, to many of the embedded systems developed for ESD. A coordinated effort with SD to collect the missing data elements and to establish a data sharing agreement for the future would result in substantial benefits to ESD.

Finally, there are many other organizations within the DoD who are involved in software development and are in the process of developing software data bases. Data sharing agreements with these organizations can yield additional relevant data points for the ESD data base.

4. ADVANCING THE STATE-OF-THE-ART OF SOFTWARE COST ESTIMATION

The development of an ESD software cost data base is only the first step necessary to enhance the software cost estimation capability at ESD. These data can be used to fine tune existing cost estimating techniques and to develop new tools based on ESD experience.

4.1 DATA BASE AUTOMATION

A data base with as many distinct elements per project as the one developed under this effort becomes very unwieldy even when it contains a small number of projects. As the data base grows, the update and maintenance tasks require more effort and data retrieval becomes a tedious, time-consuming task. Manual transfer of data into model calibration or model development format can introduce errors which may invalidate the results. Automation of the data base can eliminate many of these problems and significantly increase the useability of the data base.

The automated data base should have the following capabilities:

- User friendly data entry with error checking
- Data base editing and update

- Automatic data base back-up
- Keyword search and retrieval
- Input file generation for statistical packages and software cost estimating models.

The keyword search and retrieval system coupled with the ability to generate input files to models will minimize the resources required to assemble homogeneous groups of data for model validation, calibration, and development.

4.2 COST MODEL VALIDATION AND CALIBRATION

Because cost models, such as PRICE-S and JS-1, are based on specific sets of data that may not be representative of ESD software developments, they should be validated for use by ESD. Using the data base developed under this effort, ESD should evaluate the COCOMO, JS-1, SLIM, and PRICE-S as predictors of cost for software developments representative of ESD experience.

Initially, the ESD data base should be compared with the data used to develop the cost models which are to be validated. The major similarities and differences should be identified to facilitate analysis of model outputs. The input parameters for these models are extracted from the historical data so that the predicted cost and schedule can be compared with the actual cost and schedule data. The results are then analyzed to identify the cause of any anomalies and determine the accuracy of the models. The time-phased resource expenditure data generated by the models should also be compared with the historical data to determine its validity.

If a model is found to be suitable for estimating ESD software development cost and schedule, it is calibrated for homogeneous subsets of the ESD data base.

4.3 ESD-UNIQUE MODEL DEVELOPMENT

The validation effort described in the previous section may result in the identification of serious deficiencies in the commercially available models. If the calibration feature of the models cannot compensate for the differences between ESD developments and the projects represented in the model data bases, the development of a model unique to ESD is required.

Development of the model begins with a statistical analysis of the data base to identify the best predictors of electronics system software development costs. Selection of model variables would concentrate on those data items that are available to the estimator or can reasonably be predicted from historical data. Anyone who has ever developed an estimate is sensitive to problems of data availability to support estimates.

Based on the results of the data base analysis, a comprehensive software cost estimating model would be developed with the following features:

- Cost estimating relationships which account for variations in software development cost due to the characteristics and requirements of the system, to the expected development team profile, and to the development environment
- Historically-based resource expenditure profiles

- Impact assessment of resource constraints
- Technology adjustment factor
- Sensitivity analysis mechanism
- Capability to develop confidence intervals for the estimate
- Cost/schedule risk assessment
- Data base interface for size and technical definition by analogy.

With this tool, a cost estimator could take a detailed technical description of a software development program, determine the software size and technical characteristics by analogy, specify a development environment or use ESD historical environmental characteristics, and predict a software development cost and schedule. He could then perform sensitivity analysis based on different assumptions about the nature of the software, the development environment, and resource constraints. Finally, since the model is specifically tailored to ESD software, he would have greater confidence in the results of the analysis.

4.4 SOFTWARE SIZING METHODOLOGY

Delivered source lines of code continues to be the most often used metric for software cost estimating. Therefore, the quality of an estimate can be significantly improved with better sizing tools.

The ESD cost data base can be used as a verification tool for engineering estimates of software size:

- Using size ranges for categories of software for gross level confidence checks

- Using a catalog of standard software modules for ESD product categories to validate both the sizing and technical completeness of the engineering estimate.

In addition to confidence checks, the data base can be used to derive a size estimate by matching system functional requirements with the keyword classifications of the elements in the data base. This analogy technique is used at the lowest level of detail, system, CPCI, or module/unit, that the available system definition will allow.

In order to normalize the data among different programming languages and increase the effective size of the data base, the relationship between delivered source lines of code and size in memory words occupied can be used to develop conversion ratios.

4.5 SOFTWARE MAINTENANCE COST MODEL

Although this data collection effort was primarily concerned with software development cost estimating, some preliminary research into software maintenance cost estimating was conducted. Since software maintenance represents a significant portion of the total life-cycle cost of a system, a model for estimating these costs should be developed.

The following technical objectives must be achieved before a model can be developed:

- Definition of the basic elements of the software maintenance process including the software maintenance facility, the maintenance workload, the testing requirements, the configuration control and documentation requirements

- Identification of the potential variables for use in modeling these basic elements of the process

- Identification of ESD computer system characteristics which may be relevant to the maintenance process

- Postulation of the relationships among the element variables and the system characteristics/requirements.

The above activities will form the basis for establishing the maintenance cost data base requirements and appropriate work breakdown structure. Following a data collection effort at Air Force software maintenance facilities, the resulting data base can be analyzed statistically and a model can be developed based on the best predictor variables.

5. SUMMARY AND RECOMMENDATIONS

A comprehensive data collection package was developed under this effort and refined as a result of the feedback from the initial data collection. In the near-term, this data collection package can be used for obtaining information for completed or ongoing projects and for cost model input parameters.

For the long-term a formalized method for data collection is required to maintain the data base and ensure its continued usefulness. It is recommended that ESD become involved in the current Joint Logistics Commanders effort to revise many of the existing software data items and incorporate summary data formats into those data items. While these revised data items are in the review and approval cycle, the tailored data item approach should be used on software development efforts for which contracts will be awarded in the near-term.

In addition to the data that can be obtained for ESD programs, the size of the data base can be significantly increased through data sharing arrangements with other DoD agencies. The Air Force Space Division would be a good starting point.

The availability of data is the crucial first step for advancing the state-of-the-art of software cost estimation. Figure 5-1 illustrates a road map for increasing the size of the data base and using it to improve existing techniques, as well as for developing new tools. Future efforts should concentrate on:

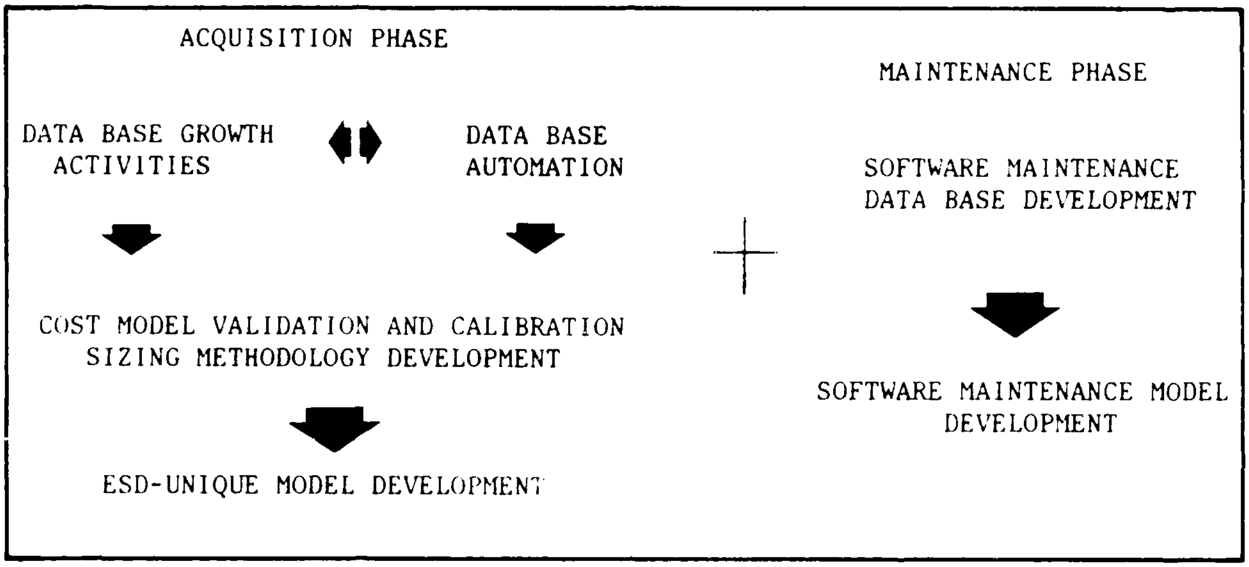


Figure 5-1   Software Life-Cycle Cost Estimation Road Map

- Data base automation
- Continued data base growth
- Validation and calibration of existing cost models
- Development of an ESD-unique cost model
- Development of software sizing tools
- Development of a software maintenance cost estimation model.

The potential offered by the data availability makes ESD a leader in the state-of-the-art of software cost estimating.

APPENDIX A

SOFTWARE COST MODEL/DATA COLLECTION FORMAT CROSS REFERENCE TABLES

The following tables cross reference the cost model input parameters to specific items on the data collection formats contained in Appendix B. Many of the items can be input directly into the cost models, while others are derived from several data items using estimator judgement. Validation and calibration activities would require the same input parameters for several homogeneous CPCIs.

TABLE A-1

COCOMO MODEL DATA REQUIREMENTS

INPUT PARAMETER	DATA COLLECTION FORMAT REFERENCE
Development Mode Selection	Project Summary Form Items 3.1, 3.2, 3.7,7 CPCI Summary Form Items 6.1A, 8 Development and Target Computer Form Items 1.7, 1.8
Software Size	CPCI Summary Form Items 8.1, 8.9 Function & Sizing Detail Form
Required Software Reliability	CPCI Summary Form Item 5
Data Base Size	CPCI Summary Form Item 8.3
Product Complexity	CPCI Summary Form Item 6
Execution Time Constraint	Development and Target Computer Data Form Items 1.4, 1.6, 1.9B
Main Storage Constraint	Development and Target Computer Data Form Items 1.2, 1.3, 1.5, 1.9, 1.10
Virtual Machine Volatility	Development and Target Computer Data Form Item 1.8
Computer Turnaround Time	Development and Target Computer Data Form Item 2.2
Analyst Capability	CPCI Summary Form Item 4.2
Applications Experience	CPCI Summary Form Item 4.1A
Programmer Capability	CPCI Summary Form Item 4.2
Virtual Machine Experience	CPCI Summary Form Item 4.1D
Programming Language Experience	CPCI Summary Form Item 4.1C
Modern Programming Practices	Project Summary Form Item 4 CPCI Summary Form Item 4.1B
Use of Software Tools	Project Summary Form Item 5 CPCI Summary Form Item 4.1E Development and Target Computer Data Form Item 2.8

TABLE A-1

COCOMO MODEL DATA REQUIREMENTS (Continued)

INPUT PARAMETER	DATA COLLECTION FORMAT REFERENCE
Required Development Schedule	Project Summary Form Item 6 CPCI Summary Form Item 3
Development Manmonths	For validation/calibration Resource Expenditure Form
Schedule Duration	For validation/calibration Project Summary Form Item 6 CPCI Summary Form Item 3

TABLE A-2

JS-1 MODEL DATA REQUIREMENTS

INPUT PARAMETERS	DATA COLLECTION FORMAT REFERENCE
Software Size	CPCI Summary Form Items 8.1, 8.9 Function & Sizing Detail Form
Interactive Environment Rating	Development and Target Computer Data Form Item 2.4
Resource Rating	Development and Target Computer Data Form Item 2.2
Tool Quality Rating	Project Summary Form Item 5 CPCI Summary Form Item 4.1E Development and Target Computer Data Form Item 2.8
Project Complexity Values	Project Summary Form Items 3.1, 3.2, 3.3, 3.4 CPCI Summary Form Items 8.4, 8.5, 8.6, 8.9 Development and Target Computer Data Form Items 1.7, 1.8
Response Requirements	CPCI Summary Form Item 8.5
Source Statement Type Mix	CPCI Summary Form Item 8.6
Development Factor	CPCI Summary Form Item 8.9
Special Display Requirements	CPCI Summary Form Item 8.7
Detailed Definition of Operational Requirements	Project Summary Form Item 8 Estimator Judgement
Real Time Operation	CPCI Summary Form Item 8.5A
CPU Time Constraint	Development and Target Computer Data Form Item 1.11
CPU Memory Constraint	Development and Target Computer Data Form Item 1.10
First Software Developed on CPU System	CPCI Summary Form Item 4.1D
Concurrent ADP Hardware Development	Development and Target Computer Data Form Items 1.7, 1.8

TABLE A-2

JS-1 MODEL DATA REQUIREMENTS (Continued)

INPUT PARAMETERS	DATA COLLECTION FORMAT REFERENCE
Developer Using Remote Computer	Development and Target Computer Data Form Items 2.6, 2.7
Development at Operational Site	Development and Target Computer Data Form Items 2.3, 2.9
Development Computer Different From Target Computer	Development and Target Computer Data Form Items 1.1, 2.1
Development at Multiple Sites	Development and Target Computer Data Form Items 2.5, 2.6, 2.7
First Use of Programming Language	CPCI Summary Form Items 4.1C, 8.8
System Type	CPCI Summary Form Item 5
Documentation Type	CPCI Summary Form Item 7

TABLE A-3

SLIM MODEL DATA REQUIREMENTS

INPUT PARAMETERS	DATA COLLECTION FORMAT REFERENCE
Software Size	CPCI Summary Form Items 8.1, 8.9 Function & Sizing Detail From
Percent of Development On-line/Interactive	Development and Target Computer Data Form Items 2.2, 2.3E
Proportion of Development Computer Dedicated to Effort	Development and Target Computer Data Form Item 2.9
Proportion of Development Computer Used for Other Work	Development and Target Computer Data Form Item 2.9
Proportion of System Coded in Higher Order Language	CPCI Summary Form Item 8.8
Primary Language Used	CPCI Summary Form Item 8.8
Software System Type	Project Summary Form Items 3.1, 3.3, 3.4 CPCI Summary Form Items 2, 8.4, 8.5
System Level	Project Summary Form Items 3.1, 3.2, 3.3, 3.4 CPCI Summary Form Items 2, 8.9
Target Machine Memory Utilization	Development and Target Computer Data Form Items 1.2, 1.3, 1.5, 1.9A, 1.10
Proportion of Real-Time Code	CPCI Summary Form Item 8.5
Modern Programming Practice Usage	Project Summary Form Item 4 CPCI Summary Form Item 4.1B
Personnel Experience	CPCI Summary Form Items 4.1A, 4.1C, 4.1D, 4.1E, 4.2
State of Technology Factor	Derived By Calibration Using CPCI Summary Form Items 3, 8.1 Resource Expenditure Form

TABLE A-4

PRICE-S MODEL DATA REQUIREMENTS

INPUT PARAMETERS	DATA COLLECTION FORMAT REFERENCE
Project Magnitude	CPCI Summary Form Items 8.1, 8.8, 8.9 Function & Sizing Detail Form Development and Target Computer Data Form Item 1.2
Project Application	CPCI Summary Form Item 8.4
Level of New Design and Code	CPCI Summary From Item 8.9
Resource	Derived by calibration using Project Summary Form Item 8 CPCI Summary Form Items 3, 4.1, 4.2, 8.1, 8.4, 8.8, 8.9 Development and Target Computer Data Form Items 1, 2.5
Utilization	Development and Target Computer Data Form Items 1.2, 1.3, 1.4, 1.5, 1.6, 1.9, 1.10, 1.11
Platform	Project Summary Form Items 3.1, 3.2, 3.3, 3.4, 3.7
Complexity or Schedule	Project Summary Form Item 8 CPCI Summary Form Items 3, 4.1, 4.2 Development and Target Computer Data Form Items 1.7, 1.8, 2.5, 2.6, 2.7
New Design	CPCI Summary Form Item 8.9
New Code	CPCI Summary Form Item 8.9
Maximum Manloading	CPCI Summary Form Items 4.3, 4.4
Mix	CPCI Summary Form Item 8.4
Interface Types	Project Summary Form Item 3.2
Interface Quantities	Project Summary Form Item 3.2

APPENDIX B
DATA COLLECTION PACKAGE

This appendix contains the ESD software cost data base data collection package, which consists of a comprehensive set of instructions containing reference tables and a glossary. Five data collection formats are also included.

SOFTWARE DATA COLLECTION FORM INSTRUCTIONS

INTRODUCTION

The objective of this data collection effort is to develop a data base of software cost, schedule, sizing, and technical information for use in cost model validation and calibration, for software sizing, and for cost model development.

There are five different forms used to collect the required data:

Software Development Project Summary Data Form - collects data at the project level to define the project functional and technical characteristics, the development tools and methods available, the project schedule, the documentation required, and the change history. It is prepared for each project for which data is provided.

Development and Target Computer Data Form - collects data for each target computer (operational host) in the system and for the development computer on which the corresponding operational software is developed.

Computer Program Configuration Item Summary Data - collects information at the CPCI level to define the development schedule, the personnel characteristics and constraints, the CPCI size and characteristics, documentation requirements, and the failure/error history during the development. This form is prepared for every CPCI identified on the software development project summary form.

Resource Expenditure Data Form - collects timephased manpower utilization data at the lowest level available for the project.

Computer Program Configuration Item Function and Sizing Data Detail Form - collects software size, function and programming language information at the lowest level available for each CPCI listed on the software development project summary data form.

Detailed instructions for completing the forms are included in the following sections. Attachment A is a glossary of terms used in the forms. Attachment B is a recommended work breakdown structure for software development projects.

SOFTWARE DEVELOPMENT PROJECT SUMMARY DATA

1. Project Name: ____________________ Date: ______
2. Development Contractor/Organization: ____________________
3. Project Description
   - 3.1 Mission Description: ____________________
   ____
   ____
   - 3.2 Major Hardware Interfaces: ____________________
   ____
   ____
   - 3.3 Major System Functions: ____________________
   ____
   ____
   - 3.4 Major Software Functions: ____________________
   ____
   ____
   - 3.5 Number of CPCIs: ________
   - 3.6 CPCI Names: ____________________
   ____
   ____
   - 3.7 System User:
     - Development Contractor ___ Other Commercial Company ___
     - Department of Defense ___ Other Government Agency ___
4. Development Methodologies Used
   - 4.1 Specification:
     - Functional ___ Procedural ___ English ___ Other: ________
   - 4.2 Design:
     - Top Down ___ Bottom Up ___ Iterative Enhancement ___
     - Hardest First ___ None ___ Other: ____________
   - 4.3 Development:
     - Top Down ___ Bottom Up ___ Iterative Enhancement ___
     - Hardest First ___ None ___ Other: ____________
   - 4.4 Coding:
     - Simulating Construct ___ Structured Code ___ None ___
     - Other: ____________

4.5 Testing:

Top Down(stubs) ___ Bottom Up(drivers) ___

Specification Driven ___ Structure Driven ___

None ___ Other: ______________________________

4.6 Validation/Verification(Inspection):

Peer Review ___ Walk Throughs ___ Proof ___

None ___ Other: ______________________________

4.7 Formalisms:

Program Design Language(Specify): ______________________________

HIPO Charts ___ Flowcharts ___ Chapin Charts ___ HOS ___

Other: ______________________________

5. Software Development Tools Used:

Assembler ___	Basic Linker ___
Basic Monitor ___	Batch Debug Aids ___
Higher Order Language Compiler ___	Macro Assembler ___
	Simple Overlay Linker ___
Basic Source Editor ___	Language Independent Monitor ___
Basic Library Aids ___	Basic Data Base Aids ___
Real-time or Timesharing Operating System ___	Extended Overlay Linker ___
	Database Management System ___
Interactive Debug Aids ___	Simple Programming Support Library ___
Interactive Source Editor ___	Virtual Memory Operating System ___
Database Design Aid ___	Simple Program Design Language ___
Performance Measurement And Analysis Aids ___	Programming Support Library With Basic Configuration Management Aids ___
Set-Use Static Analyzer ___	Control Flow Static Analyzer ___
Basic Text Editor & Manager ___	Program Flow and Test Case Analyzer ___
File Manager ___	Full Programming Support Library ___
Documentation System ___	Project Control System ___
Requirements Specification Language and Analyzer ___	Extended Design Tools ___
	Automated Verification System ___
Fault Report System ___	Crosscompilers ___
Instruction Set Simulators ___	Display Formatters ___
Data Entry Control Tools ___	Communications Processing Tools ___
Conversion Aids ___	Structured Language Tool ___
Other: ______________________________	Other: ______________________________
Other: ______________________________	Other: ______________________________

6. Development Schedule

Project Milestone	Original Date	Actual Date	Estimated Date
Contract Award			
System Requirements Review			
System Design Review			
Preliminary Design Review			
Critical Design Review			
Preliminary Qualification Test			
Formal Qualification Test			
Start CPCI Integration Into System			
Complete CPCI Integration into System			
Start Development Test & Evaluation			
Complete Development Test & Evaluation			
Start Initial Operational Test & Eval			
Complete Initial Operational Test & Eval			
Functional Configuration Audit			
Physical Configuration Audit			
Formal Qualification Review			
System Delivery			

7. Documentation

Document Title	# of Pages	Est'd or Act'l	
System Engineering Management Plan			
Computer Program Development Plan			
System Test Plan			
Other:			
Other:			
Other:			
Other:			

8. Software Change History

Development Phase	# of Changes Approved	Est'd DSLOC Change +/-	Est'd Manpower Change +/-
Preliminary Design (Contract Award to PDR)	________	________	________
Detailed Design (PDR to CDR)	________	________	________
Code & Debug (CDR to Test & Integ Start)	________	________	________
Test & Integration (Test & Integ Start to FQT)	________	________	________
System Test/IOC (FQT to Contract End)	________	________	________

SOFTWARE DEVELOPMENT PROJECT SUMMARY DATA FORM INSTRUCTIONS

Item 1: Enter the name of the project and the date on which this form is being prepared.

Item 2: Identify the company or organization which is actually performing the software design and development.

Item 3.1: Briefly describe the overall mission or purpose of the system for which the software is being developed.

Item 3.2: Identify the major hardware components with which the software will interface, for example, radars, communications equipment, sensors, other embedded computer systems, etc.

Item 3.3: List the major functions performed by the system.

Item 3.4: List the major functions performed by the software.

Item 3.5: Identify the number of Computer Program Configuration Items (CPCIs) into which the system is divided.

Item 3.6: List the names of each CPCI in the system.

Item 3.7: Indicate with an X the user for whom the system is being developed.

Items 4.1-4.7: Indicate with an X all of the software development methodologies and strategies used for each activity on this project.

Item 5: Indicate with an X all of the software development tools used on this project; use the last four items to specify other tools used which are not included in the listing.

Item 6: Enter the original schedule date for each applicable milestone (enter N/A if a milestone is not applicable). Although these milestones represent formal contractual activities in the Department of Defense software acquisition process, many non-defense projects will have milestones which are equivalent to these, e.g., contract award is equivalent to project start and critical design review is equivalent to completion of detail design. If the formal milestones are not required in the project schedule, data for equivalent activities should be used. Definitions of these milestones are provided in Attachment A of these instructions. Unless otherwise indicated, the date should reflect the activity completion date. Where available, enter the actual date of completion for the milestone; for ongoing efforts, enter the current estimate for completion of the milestone.

Item 7: Enter the number of pages for each document listed and specify any additional documentation required for the software at the project level. Indicate with an X in the appropriate column whether the page count is estimated or actual.

Item 8: Enter the number of requirements changes which occurred during each completed development phase, the net increase/decrease in the total system source instruction count and the net increase/decrease in the estimated manpower for the software development effort.

DEVELOPMENT AND TARGET COMPUTER DATA FORM

1. Target Computer

1.1 Manufacturer: ______ Model Number: ______

1.2 Main Memory Size in Words: ______ Word Size: ______ Bits

1.3 Maximum Main Memory Size: ______

1.4 CPU Processing Speed: ______

1.5 Reserve Memory Requirement: ______ %

1.6 Reserve Timing Requirement: ______ %

1.7 Concurrent Development with Software: Yes ___ No ___

1.8 Virtual Machine Volatility:

Very Low ___ Low ___ Nominal ___ High ___ Very High ___

1.9 Percent Utilization: <51% 51-70% 71-85% 86-95% >95%

A. CPU Memory

B. Execution Time

1.10 CPU Memory Constraint Evaluation:

No Memory Economy Measures Required ___

Some Overlay Use Or Segmentation Required ___

Extensive Overlay And/Or Segmentation Required ___

Complex Memory Management Economy Measures Required ___

1.11 CPU Time Constraint Evaluation:

No Software is CPU Time Constrained ___

<25% of Source Code is Time Constrained ___

<50% of Source Code is Time Constrained ___

<75% of Source Code is Time Constrained ___

1.12 CPCIs Hosted on This Computer:

2. Development Computer

2.1 Same as Target Computer: Yes ____ No ____

If No, Manufacturer: ______________ Model Number: ____________

Difference Between Development and Target Computer: ___________

____

____

2.2 Turnaround Time:

Low (interactive, specify): ________ Nominal (<4hrs) ____

High (4-12hrs) _____ Very High (>12hrs) _____

2.3 Percentage of Source Instructions Developed Using Each of the Following Access Modes (Total=100%):

A. Batch ______%

B. Dedicated Processor ______%

C. Test Bed with High Priority ______%

D. Test Bed with Low Priority ______%

E. Interactive ______%

F. Other: ____________________ ______%

2.4 For Interactive Development, Indicate the Average Number of Software Engineers/Programmers per Terminal: ___________

2.5 Number of Development Sites ________

2.6 Development Site Locations: ______________________________

____

____

2.7 Development Computer Location(s): __________________________

____

____

2.8 Software Development Tool Usage:

Very Low Low Nominal High Very High

2.9 Development Computer Resource Availability:

DEVELOPMENT AND TARGET COMPUTER DATA FORM INSTRUCTIONS

Item 1.1: Identify the manufacturer of the operational system for which this software is being developed. If the computer is an off-the-shelf item, enter the model number.

Item 1.2: Enter the main memory size in words and indicate the word size in bits. For ongoing projects, this should reflect the current configuration of the computer. For completed projects, this entry should reflect the delivered configuration of the computer.

Item 1.3: Enter the maximum main memory size in words that can be attained without major modification to the current or delivered computer configuration.

Item 1.4: Indicate the CPU processing speed in instructions per second.

Item 1.5: Enter the percent of Item 1.2 which must be left available for expansion/growth after system delivery.

Item 1.6: Enter the percent of the total processing time which must be left available for expansion/growth after system delivery.

Item 1.7: Indicate with an X whether the target virtual machine is being developed concurrently with the software.

Item 1.8: Using the following criteria, indicate with an X the degree of volatility in the design of the virtual machine:

Very Low = No major changes; one minor change every 12 months

Low = Major changes every 12 months; minor changes every month

Nominal = Major changes every 6 months; minor changes every 2 weeks

High = Major changes every 2 months; minor changes every week

Very High = Major changes every 2 weeks; minor changes every 2 days

Item 1.9A: Indicate with an X the maximum percentage of main storage used by any group of CPCIs operating concurrently.

Item 1.9B: Indicate with an X the maximum percentage of processing time used by any group of CPCIs executing concurrently.

Item 1.10: Indicate with an X the level of memory conservation measures required to satisfy the reserve memory requirement in Item 1.5.

Item 1.11: Indicate with an X the percentage of the software that requires special coding effort to enhance timing performance.

Item 1.12: Enter the names of the CPCIs which are hosted in this computer.

Item 2.1: Indicate with an X whether the development computer is the same as the target operational computer. If the computers are different, identify the manufacturer and

model number of the development computer. Describe any differences between the two computers which would affect the software development effort, e.g., different operating systems, computers, compilers, main memory and timing constraints. Is development of a target computer emulator required?

Item 2.2: Indicate with an X the average turnaround time for the development computer. If a rating of low applies, specify the approximate response time experienced on the system per computing job, e.g., a unit test or compile.

Item 2.3: Indicate the percentage of source instructions developed using the specified access modes. Specify any other mode used and its percentage of utilization.

Item 2.4: For terminals which are readily accessible to members of the development team, indicate the average number of software engineers and programmers per terminal.

Item 2.5: Enter the number of individual sites where this software is being developed. Indicate any site that is working as a subcontractor to the development contractor.

Item 2.6: Identify the geographic location (city and state) of each of the development sites.

Item 2.7: Specify the location of the development computers used to develop this software by each of the development sites.

Item 2.8: Specify the degree to which the development tools available within the development contractor's organization are used in the development of this software.

Item 2.9: Indicate the percentage of the total development computer capacity that is available for work on this project. This percentage should reflect the impact of operational uses or other developments competing for the use of this resource.

COMPUTER PROGRAM CONFIGURATION ITEM SUMMARY DATA

1. CPCI Name: ____________

2. Functional Description: ________________________________

____

____

3. Schedule

3.1 Milestone Data

CPCI Development Milestones	Original Date	Actual Date	Estimated Date
Design Start			
Preliminary Design Review			
Development Specification Approval			
Critical Design Review			
Start Coding/Debugging			
Complete Coding/Debugging			
Start Informal Integration & Test			
End Informal Integration & Test			
Preliminary Qualification Test			
Formal Qualification Test			
Product Specification Approval			
Functional Configuration Audit			
Physical Configuration Audit			

3.2 Schedule Acceleration/Stretchout Assessment:

Below 75%   75-85%   86-130%   131-160%   160%

4. Personnel

4.1 Average Experience

	1mo	1-4mos	4-12mos	1-3yrs	3-6yrs	_6yrs
A. Application Area						
B. Techniques Used						
C. Languages Used						
D. Virtual Machine						
E. Support Software/Tools						

4.2 Average Personnel Quality Percentile:

0-15% ___ 16-35% ___ 36-55% ___

56-75% ___ 76-90% ___ 90-100% ___

4.3 Manpower Availability: ________%

4.4 Peak Manloading: __________

5. Reliability Requirement:

Very Low ___ Low ___ Nominal ___ High ___ Very High ___

6. Complexity:

Very Low ___ Low ___ Nominal ___

High ___ Very High ___ Extra High ___

7. Quality of Specification

Very Precise ___ Precise ___ Imprecise ___

8. Size

8.1 Deliverable Lines of Source Code Excluding Documentation: __________

8.2 Lines of Source Code Documentation: __________

8.3 Data Base Size in Bytes or Characters: __________

8.4 Size Breakdown by Operation As a Percent of Item 8.1(Total = 100%):

A. Data Storage & Retrieval ______%

B. Online Communications ______%

C. Real-Time Command & Control ______%

D. Interactive Operations ______%

E. Mathematical Operations ______%

F. String Manipulation ______%

G. Operating Systems ______%

H. Other: __________________ ______%

I. Other: __________________ ______%

8.5 Operational Response Requirements Distribution As a Percent of Item 8.1(Total = 100%):

A. Real-Time ____% B. On-Line ____%

C. Time-Constrained ____% D. Non-Time Critical ____%

8.6 Source Statement Type Mix As a Percent of Item 8.1(Total = 100%):

A. Logical ____% B. Command ____% C. Mathematical ____%

D. Data Manipulation ____% E. Data Declaration ____%

8.7 Special Display Requirements:

Simple Input/Output ___ User Friendly ___

Interactive ___ Complex Requirements/Severe Impact ___

8.8 Languages Used as a Percent of Item 8.1(Total = 100%):

A. Language: ____________ Percentage: ________%

B. Language: ____________ Percentage: ________%

C. Language: ____________ Percentage: ________%

D. Language: ____________ Percentage: ________%

8.9 Reusable Code From Similar Projects

Project	# of DSLOC	% of Modification Design	Code	Required Integration
		%	%	%
		%	%	%
		%	%	%
		%	%	%

9. Documentation

Document title	# of Pages	Est'd or Act'l
CPCI Development Specification		
CPCI Product Specification		
Test Plan		
Test Procedures		
Test Report		
User's Manual		
Operator's Manual		
Other:		
Other:		
Other:		

10. Software Failure History

Development Phase	# of Software Failures/Errors
Preliminary Design(Contract Award to PDR)	______
Detailed Design(PDR to CDR)	______
Code & Debug(CDR to Test & Integ Start)	______
Test & Integration(T & I Start to FQT)	______
System Test/IOC(FQT to Contract End)	______

11. Other Factors or Characteristics: ______

COMPUTER PROGRAM CONFIGURATION ITEM SUMMARY DATA FORM INSTRUCTIONS

Item 1: Enter the name of the CPCI for which this form is being prepared.

Item 2: Enter a brief description of the major functions performed by this CPCI.

Item 3.1: Enter the original schedule date for each applicable milestone (enter N/A if a milestone is not applicable). If the milestones are not established for this project, use the schedule data for equivalent activities to complete this item. Formal milestone definitions are included in Attachment A. Unless otherwise indicated, the date should reflect the activity completion date. Where available, enter the actual date of completion for the milestone; for ongoing efforts, enter the current estimate for completion of the milestone.

Item 3.2: Indicate with an X the degree of schedule acceleration or stretchout that the original schedule dates in Item 3.1 represent relative to the normal time required to develop this CPCI. For example, if the specified schedule is 24 months and the normal development time is estimated at 30 months, the schedule acceleration/stretchout is 80%.

Item 4.1: For each of the five areas listed, indicate the average experience at the start of this project of the development personnel working on this CPCI. Item 4.1A refers to experience with other projects having similar functions and interfaces. Item 4.1B refers to experience with the develop ment tools and methods used on this CPCI. Item 4.1C refers to experience with the programming languages used on

this CPCI. Item 4.1D refers to experience with the development and target computer hardware, operating systems and architecture. Item 4.1E refers to experience with the support software and automated development tools, e.g., program design languages, debuggers, etc., used in the development of this CPCI.

Item 4.2: Indicate with an X the capability of the analysts and programmers who are working on this CPCI in terms of percentiles with respect to the overall industry population of programmers/designers. This rating should be based on aptitude for programming/designing software, efficiency and thoroughness, and ability to communicate and cooperate. This rating should reflect the capability of the personnel as a team rather than individuals.

Item 4.3: Indicate as a percentage the degree to which manpower loading levels are constrained by personnel availability or budget limitations. An entry of 85% indicates that the attainable manpower loading was 15% less than the required level. If there are no manloading constraints, enter 100%.

Item 4.4: For completed developments enter the peak manloading level, i.e., the largest number of software engineers and programmers at a point in time, attained during the development of this CPCI. For ongoing developments enter the current estimate of the maximum manloading required.

Item 5: Indicate with an X the level of reliability required for this CPCI using the following impact criteria:

> Very Low = The impact of a software failure is simply the inconvenience caused by the requirement to fix the software. Typical examples are a demonstration prototype of a voice typewriter or an early feasibility phase software simulation model.

Low = The effect of the failure is a small, easily recoverable loss to the users. Typical examples are a long range planning model or a climate forecasting system.

Nominal = The effect of software failure is a moderate loss to users, but a situation from which one can recover without extreme penalty. Typical examples are management information systems or inventory control systems.

High = The effect of the software failure can be a major financial loss or a massive human inconvenience. Typical examples are banking systems and electric power distribution systems.

Very High = The effect of software failure can be the loss of human life. Examples are military command and control systems or nuclear reactor control systems.

Item 6: Indicate with an X the level of complexity of this CPCI using the criteria in Table B-1 by matching the characteristics for each type of processing performed by this CPCI.

Item 7: Indicate with an X the degree of precision in the development specification using the following criteria:

Very precise = No additional analysis is needed to develop detail design

Precise = Only minor details must be worked out to develop detail design

Imprecise = Significant additional analysis is required to develop detail design.

Item 8.1: Enter the total deliverable lines of source code for this CPCI. Do not include lines which are entirely documentation, such as, comments or source instructions from unmodified utility software. This line count should include job control language instructions, format statements, and data declarations as well as logic control instructions.

TABLE B-1
SOFTWARE COMPLEXITY CRITERIA

TYPE RATING	CONTROL OPERATIONS	COMPUTATIONAL OPERATIONS	DEVICE-DEPENDENT OPERATIONS	DATA MANAGEMENT OPERATIONS
Very low	Sequenced code with a few non-nested SP operators: DOs CASEs, IFTHEN-ELSEs. Simple predicates	Evaluation of simple expressions, e.g. A=B+C*(D-E)	Simple read, write statements with simple formats	Simple arrays in main memory
Low	Straightforward nesting of SP operators. Mostly single predicates	Evaluation of moderate level expressions e.g., D=SQRT (B**2-4.*A*C)	No cognizance needed of particular processor or I/O device characteristics. I/O done at GET/PUT level, no cognizance of overlap	Single file subsetting with no data structure changes, no data edits, no intermediate files
Nominal	Mostly simple nesting. Some intermodule control. Decision tables	Use of standard math and statistical routines. Basic matrix and vector operations	I/O processing includes device selection Status checking and error processing	Multiple input and single file output. Simple structural changes simple edits
High	Highly nested SP operators with many compound predicates. Queue and stack control. Considerable intermodule control	Basic numerical analysis: multivariate interpolation, ordinary differential equations. Basic truncation roundoff concerns	Operations at physical I/O level (physical storage address translations, seeks, reads, etc.) Optimized I/O overlap	Special purpose subroutines activated by data stream contents. Complex data restructuring at record level

TABLE B-1

SOFTWARE COMPLEXITY CRITERIA (Continued)

TYPE RATING	CONTROL OPERATIONS	COMPUTATIONAL OPERATIONS	DEVICE-DEPENDENT OPERATIONS	DATA MANAGEMENT OPERATIONS
Very High	Reentrant and recursive coding. Fixed-priority interrupt handling	Difficult but structured numerical analysis: near-singular matrix equations, partial differential equations	Routines for interrupt diagnosis, servicing, masking. Communication line handling	Generalized parameter-driven file structuring routine. File building, command processing, search optimization
Extra High	Multiple resource scheduling with dynamically changing priorities. Microcode level control	Difficult and unstructured numerical analysis: highly accurate analysis of noisy stochastic data	Device timing-dependent coding, micro-programmed operations	Highly coupled, dynamic relational structures. Natural language data management

Item 8.2: Enter the total lines of source code documentation delivered with the source code for this CPCI.

Item 8.3: Enter the size of the data base to be developed with this CPCI and delivered as part of the system in bytes.

Item 8.4: Enter the percent of the deliverable lines of source code that performs each of the categories of operation defined below:

Data Storage and Retrieval - Operation of data storage devices, data base management, secondary storage handling, data blocking and deblocking, hashing techniques. Primarily hardware oriented code.

On-line Communications - Including machine-to-machine communications with queuing allowed. Timing restrictions not as severe as with real-time command and control.

Real-time Command and Control - Machine-to-machine communications under tight timing constraints. Queuing not practicable. Heavy hardware interface. Strict protocol requirements.

Interactive Operations - Real-time man/machine interfaces. Human engineering considerations and error protection are very important.

Mathematical Operations - Routine mathematical applications with no overriding constraints.

String Manipulation - Routine applications with no overriding constraints. Not oriented towards mathematics. Typified by language compilers, sorting, formatting, buffer manipulation, etc.

Operating Systems - Task management. Memory management. Heavy hardware interface. Many interactions. High reliability and strict timing requirements.

Other - Specifically identify any unique operations not included in the above categories.

Item 8.5: Indicate with an X the response mode required in the operational system using the following guidelines:

Real-time - The software must complete processing in response to an event prior to the occurrence of the next event. Arrival of the data and the occurrence of events is not under the control of the software and extra effort in the design, test and implementation of the software is required to satisfy time and processing requirements.

On-line - Software in this category must respond within a human compatible time frame, usually within a few seconds. Also requires additional development effort, but not the extra level required for real-time software.

Time-constrained - Software in this category must complete processing within a specified time frame which is not as restrictive as real-time or on-line requirements. Time lines are in the order of minutes or hours; sometimes a clock time is specified for completion of processing.

Non-time Critical - There is no time constraint for completion of processing for this category of software.

Item 8.6: Enter the percentage of the delivered lines of source code for this CPCI for each of the statement types listed using the following guidelines:

Logical - statements which control the execution sequences in the program and include constructs such as IF-THEN-ELSE, DO WHILE, DO UNTIL, CASE, GO TO or CALL.

Command - statements which direct the system software to perform specific functions or to create the environment required to support the software. These statements are generally written in a language specific to the computer hardware.

Mathematical - statements which perform computations. This category includes coded equations for algorithms, vector algebra, modeling, index computation, etc.

Data Manipulation - statements which perform input and output, as well as the storage, movement and modification of data. Format statements are also included.

Data Declaration - statements which are non-executable and define the characteristics and values of the data contained in the program.

Item 8.7: Indicate with an X the level of display interaction required for the user interface with this CPCI using the following guidelines:

Simple Input/Output - No special considerations or requirements implemented to enhance user interface.

User Friendly - Special display formatting, e.g., menus, with extensive diagnostic and input or processing error handling capabilities.

Interactive - Advanced features such as light pen or touch-sensitive displays for user interface.

Complex Requirements/Severe Impact - Two dimensional projection of solid figures, hidden line processing in line-of-sight projections, earth projections from space for weather prediction and resource mapping and integrated circuit layout.

Item 8.8: Identify the programming languages in which this CPCI is written and indicate the percentage of the delivered source lines coded in the language.

Item 8.9: Identify any previously developed code which has been adapted for this CPCI. Include the name of the project for which the code was developed and the number of delivered source lines of code (DSLOC) which were adapted. Enter the percentage of the original design which was modified and the percentage of code which was rewritten. Finally, indicate the approximate percentage of effort required to integrate and test the adapted software compared to the normal amount required for a new development of a comparable size and difficulty.

Item 9: Enter the page count for each document listed and specify any additional documentation required for this CPCI. *Indicate with an X in the appropriate column whether* the page count is estimated or actual.

Item 10: Enter the number of software failures, design errors and coding errors discovered during each of the five development phases.

Item 11: Describe any other factors or characteristics of this CPCI and the development environment which affected the number of delivered source lines of code or the resources expended in the development of this CPCI. Identify any features of this development which make it unique relative to other developments.

RESOURCE EXPENDITURE DATA

1. Project Name: ____________________

2. Latest Month of Actuals: __________ 3. Units of Manpower: __________

WBS ELEMENT OR CPCI / MONTHS AFTER CONTRACT AWARD					
1					
2					
3					
4					
5					
6					
7					
8					
9					
10					
11					
12					
13					
14					
15					
16					
17					
18					
19					
20					
21					
22					
23					
24					
25					

MAC \ WBS ELEM/CPCI					
26					
27					
28					
29					
30					
31					
32					
33					
34					
35					
36					
37					
38					
39					
40					
41					
42					
43					
44					
45					
46					
47					
48					
49					
50					
51					
52					
53					
54					
55					
56					
57					
58					
59					
60					

RESOURCE EXPENDITURE DATA FORM INSTRUCTIONS

This form is designed to collect time-phased manpower data for the software development project at the lowest level of detail available. Attach a copy of the cost/work breakdown structure used to collect manpower data for software activities on this contract/development project.

Item 1: Enter the project name.

Item 2: Enter the latest month after contract award or project start for which actual manpower data is available. This number should reflect the months after the date for contract award entered in Item 6 for the contract award milestone on the Software Development Project Summary Data Form.

Item 3: Enter the units of measure used for the manpower figures, that is, manhours, mandays, manmonths or manyears. Indicate the number of hours that the unit is based on, if you are not entering manhours.

In the top row of the table enter the name of the cost/work breakdown structure element or computer program configuration item for which you have manpower data. Include any of the software related work breakdown structure elements in Attachment B for which you have data. In each of the subsequent rows enter the manpower expended during the month after contract award indicated in the left hand column. Space is provided for up to five years of data. For ongoing projects enter the latest estimate of resource requirements for those months for which actual data is not available.

COMPUTER PROGRAM CONFIGURATION ITEM FUNCTION AND SIZING DATA DETAIL

CPCI NAME: ______________________

CPC NAME	MODULE OR UNIT NAME	REUSED CODE LITTLE OR NO MODIFICATION	REUSED CODE EXTENSIVE MODIFICATION	NEW CODE	FUNCTION	SIZING		PROGRAMMING LANGUAGE
						# OF DISLOC	# OF WORDS	

COMPUTER PROGRAM CONFIGURATION ITEM FUNCTION AND SIZING DATA DETAIL FORM INSTRUCTIONS

Enter the name of the CPCI for which the information is being furnished in the space provided. For each module or unit in the CPCI, identify computer component to which it belongs, the function index number from Table B-2, the module size in delivered source lines of code and number of main memory words occupied, and the programming language used. Likewise, indicate with an X in the appropriate column whether the code is reused code with little or no modification, reused code with extensive modification, or entirely new code. If information is not available at the module or unit level, provide it at the next highest level available, i.e., CPC level. If the CPCI does not use all of the intermediate levels, leave the unused column blank. In the event that none of the functions in Table B-2 adequately describes a particular CPCI element, enter a brief statement of the element function in the appropriate column.

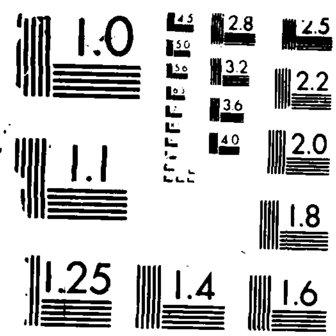
TABLE B-2

SOFTWARE FUNCTION CATEGORIES

TYPE	CATEGORY	INDEX	FUNCTION
Operational	Displays	1.1	Avionics
		1.2	Command, Control, Communications
		1.3	Other
	Avionics	2.1	Mission Planning
		2.2	Navigation
		2.3	Aircraft Steering & Flight Control
		2.4	Sighting, Designation & Location Determination
		2.5	Weapon Delivery
		2.6	Electronic Countermeasures
		2.7	Other
	Command, Control, & Communications	3.1	Network Monitoring
		3.2	Network Control & Switching
		3.3	Sensor Control
		3.4	Signal Processing
		3.5	Message Processing
		3.6	Message Distribution
		3.7	Message Logging & Retrieval
		3.8	Data Reduction
		3.9	Other
	Executive	4.1	Computer Resource Management
		4.2	Computer Operator Interface
		4.3	Other Terminal Operator Interface
		4.4	Special Device Interface
		4.5	Other Input or Output
		4.6	Error Handling [illegible] Recovery
		4.7	Multicomputer [illegible]
		4.8	Performance [illegible]
		4.9	[illegible]

AD-A186 587 SOFTWARE DATA BASE DEVELOPMENT VOLUME 1(U) ANALYTIC SCIENCES CORP READING MA A C NAJBERG 25 JUN 84 TR-4612-5-1-VOL-1 ESD-TR-87-166-VOL-1 2/2

UNCLASSIFIED F33657-82-D-0253/0014 F/G 12/5 NL



MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS 1963-A

TABLE B-2
SOFTWARE FUNCTION CATEGORIES (Continued)

TYPE	CATEGORY	INDEX	FUNCTION
Operational (Continued)	Data Base	5.1	On-Line Retrieval & Output
		5.2	On-Line Initialization & Updating
		5.3	Other
	Training	6.1	Control of Exercise Sequencing
		6.2	Operator Performance Data Collection
		6.3	Other
	On-Line Equipment Diagnostic	7.1	System Readiness Test
		7.2	Computer Diagnostic
		7.3	Memory Diagnostic
		7.4	Display Diagnostic
		7.5	Switch/Indicator Panel Diagnostic
		7.6	I/O Diagnostic
		7.7	Mode Diagnostic
		7.8	Other
Support	Operating System	8.1	Computer Resource Management
		8.2	Computer Operator Interface
		8.3	Terminal Operator Interface
		8.4	Input or Output
		8.5	Error Handling/Reconfiguration/ Recovery
		8.6	Performance Monitoring & Data Collection
		8.7	Other
	Equipment Maintenance	9.1	Off-Line Computer Diagnostics
		9.2	Other

TABLE B-2
SOFTWARE FUNCTION CATEGORIES (Continued)

Type	Category	Index	Function
Support (Continued)	Software Development	10.1	Higher Order Language Compiler
		10.2	Assembler
		10.3	Debugger
		10.4	Loader or Editor
		10.5	Other
	Off-Line Data Base Management	11.1	Data Base Definition
		11.2	Data Base Initialization or Updating
		11.3	Retrieval & Output Formatting
		11.4	Data Base Restructuring
		11.5	Off-Line Data Base
		11.6	Other
	Design	12.1	Data Base Design
		12.2	Data Base Processor Design
		12.3	Performance Simulation
		12.4	Data Reduction
		12.5	Data Analysis
		12.6	Other
	Test Software	13.1	Test Case Generation
		13.2	Test Case Data Recording
		13.3	Test Data Reduction
		13.4	Test Analysis
		13.5	Other
	Utilities	14.1	Media Conversions
		14.2	Format Translation
		14.3	Sort/Merge
		14.4	Program Library Maintenance
		14.5	Other

TABLE B-2

SOFTWARE FUNCTION CATEGORIES (Continued)

TYPE	CATEGORY	INDEX	FUNCTION
Support (Continued)	Off-Line Training	15.1	Data Reduction
		15.2	Training Analysis
		15.3	Scenario Preparation
		15.4	Other
	Project Management	16.1	Project Event Status Accounting
		16.2	Schedule Maintenance/Projection
		16.3	Financial Accounting
		16.4	Software Cost Reporting
		16.5	Hardware Cost Reporting
		16.6	Software Cost Prediction
		16.7	Hardware Cost Prediction
		16.8	Other
	Hardware Subsystem Simulations	17.1	Interfacing Hardware Simulations
		17.2	Environmental Simulations
		17.3	Operator Action Simulations
		17.4	Other

ATTACHMENT A

GLOSSARY

Application software - software that implements the operational capabilities of a system.

Assessment - a qualitative evaluation.

Compiler - a computer program that accepts a source program expressed in a higher order language as input, and produces either a machine code or assembly language representation of the source program as output.

Code walk-through - a step-by-step, detailed examination of source code by a small group of qualified personnel. Sometimes it is referred to as a peer review.

Computer data - basic elements of information used by the computer hardware in responding to a computer program.

Computer program - a series of instructions or statements in a form acceptable to an electronic computer designed to cause the computer to execute an operation or a series of operations.

Computer software - a combination of associated programs and computer program data definitions required to enable the computer hardware to perform computational or control functions. Note: this definition includes firmware.

Computer program component (CPC) - a design component of the computer program design architecture made up of units and modules implementing requirements of a computer program configuration item (CPCI).

Computer program configuration item (CPCI) - an aggregation of computer software which satisfies an end use function and is designated for configuration management.

Contractor - any organization under contract or tasking agreement with a procuring agency to perform any part of a software development effort.

Critical Design Review (CDR) - a review conducted for each configuration item when the detail design is essentially complete to determine if the detail design satisfies the requirements

established in the specification and to establish the exact interface relationships with other parts of the system.

Defense system - a system which contributes directly to the combat capability of the Department of Defense.

Demonstration - a qualification method which relies on observable operation to establish that a requirement has or has not been met.

Design walk-through - a step-by-step detailed examination of the design of the software by a small group of qualified personnel.

Development baseline - the initial approved technical documentation which defines the configuration of a CPCI during the Full-Scale Development Phase and which prescribes (1) all design characteristics of the CPCI and (2) the selected functional characteristics of the CPCI designated for software performance testing. The Developmental Baseline is under the development contractor's configuration control.

Documentation - the comprehensive written description of computer software in various formats and levels of detail that clearly define its content, composition, design, performance, testing, and use.

Embedded computer software - software which executes on computers which are (1) incorporated as integral parts, (2) dedicated to or required for the direct support of, or (3) required to upgrade or modify defense systems.

Firmware - microcode (software) which resides in computer memory that is not alterable by the computer system during program execution.

Formal Qualification Test (FQT) - a formal test conducted in accordance with approved test plans, descriptions, and procedures after a CPCI has been integrated to validate that each function of the CPCI satisfies specified software requirements and applicable interface requirements.

Formal test - a test which is conducted in accordance with test procedures approved by the procuring activity, is witnessed by an authorized representative, and is documented in a test report for procuring agency review.

Functional Configuration Audit (FCA) - the formal examination of functional characteristics test data for a configuration item to verify that the item has achieved the performance specified in its functional or allocated configuration identification.

Higher order language - a primarily machine independent language (of a higher order than assembly language) designed for ease of expression of a class of problems or procedures by humans.

Inspection - a qualification method which relies on visual examination to establish that a requirement has or has not been met.

Measurement - quantitative evaluation.

Modular - a software design characteristic which organizes the software into limited aggregates of data and contiguous code that perform complete functions and are, therefore, completely understandable by themselves.

Module - a discrete, identifiable set of instructions which are treated as an entity by the computer's operating system, and which can be executed and tested on a stand-alone basis. Equivalent to a unit.

Physical Configuration Audit (PCA) - the formal examination of the "as-coded" configuration of a CPCI against its technical documentation in order to establish the initial product configuration identification.

Precompiler - a computer program which converts programming statements which are unacceptable to the compiler into statements with acceptable syntax.

Preliminary Design Review (PDR) - a review prior to the start of the detail design process to evaluate progress and technical adequacy of the selected design approach, to determine the design compatibility with the performance requirements of the CPCI development specification, and to establish the existence and compatibility of the interfaces between the configuration item and other elements of the system.

Preliminary Qualification Test (PQT) - a test conducted during the integration of a CPCI to evaluate the performance of those CPCI functions which are critical, as determined by time-critical or performance-critical requirements. A PQT may be either formal or informal.

Product baseline - the final approved technical documentation which defines the configuration of a CPCI at the completion of software performance testing at the point where the CPCI is integrated into the system. The product baseline includes final versions of all specifications and documents from preceding baselines.

Program design language - a design tool used to facilitate the translation of system functional requirements into the elements of a program design hierarchy.

Program support library - a repository for programs and data used to facilitate the orderly development of software. The repository provides two fundamental capabilities: (1) programs and data are stored in machine readable form for computer operation and the identical information is stored in hard copy form for human comprehension, and (2) the repository contains all management data pertinent to the software development project.

Software development - the engineering process and effort that results in software, encompassing the span of time from initiation of the contracted effort through delivery to and acceptance by the procuring agency.

Software error - an occurrence, or lack thereof, during the execution of a program and attributable to the software that prevents satisfaction of the specified software requirements, fails to perform as designed, or performs a function not required and not desired.

Support software - all software used to aid the development, testing and support of applications, systems, test and maintenance, and trainer software.

Systems software - software that is used to maximize the use of computer resources at the time of use. Operating systems, executives, and data base management systems are examples of this type of software.

System Design Review (SDR) - a review conducted when the definition effort has proceeded to the point where system requirements and the design approach are more precisely defined. The review ensures that there is a technical understanding between the contractor and the procuring agency on the system segments identified in the system specification and the configuration items identified in the CI performance specifications.

System Requirements Review (SRR) - a review conducted when a significant portion of the system functional requirements have been established to determine the adequacy of the contractor's efforts in defining system requirements.

Test - a qualification method which relies on operation in an actual or simulated environment and subsequent analysis of data obtained during operation to establish that a requirement has or has not been met.

<u>Top-down design</u> - a design method in which operations are defined in a hierarchical manner from the more general to the more precise. Top-level operations are defined in relatively general terms. Operations on all levels except the bottom one are defined more precisely in terms of operations on the level immediately below.

<u>Top-down programming</u> - the implementation of code and data in a sequence which proceeds from the top level of design to the bottom level, continuously exercising the actual interfaces between program elements.

<u>Unit</u> - the lowest level logical entity specified in the detailed design which completely describes a non-divisible function in sufficient detail to allow implementing code to be produced, assembled or compiled, and tested independently of other units. Equivalent to a module.

<u>Validation of computer software</u> - the evaluation, integration, and test activities carried out at the system level to ensure that the finally developed system satisfies the user's requirements set down as performance and design criteria for the system.

<u>Verification of computer software</u> - the interactive process of determining whether the product of each step of the software development process fulfills all requirements levied by the previous step.

<u>Virtual machine</u> - the complex of hardware and software that the software being developed calls upon to accomplish its tasks.

ATTACHMENT B

SOFTWARE DEVELOPMENT PROJECT WORK BREAKDOWN STRUCTURE

The following is an illustration of a recommended work breakdown structure for a project with both hardware and software elements.

LEVEL 1	LEVEL 2	LEVEL 3	LEVEL 4
Defense System			
	Prime Mission Equipment		
		Integration and Assembly	
		Hardware Subsystem or End Item 1	
		–	
		–	
		–	
		Hardware Subsystem or End Item n	
		Software Subsystem 1	
			Subsystem Analysis & Design
			Subsystem Integration & Test
			Computer Program Configuration Item 1
			–
			–
			–
			Computer Program Configuration Item n
			–
			–
			–
		Software Subsystem n	

Level 1	Level 2	Level 3	Level 4
		Support Software	
			Computer Program Configuration Item 1
			-
			-
			-
			Computer Program Configuration Item n
	Training		
		Equipment	
		Services	
		Facilities	
	Peculiar Support Equipment		
		Organizational	
		Intermediate	
		Depot	
	System Test & Evaluation		
		Development Test & Evaluation	
		Operational Test & Evaluation	
		Mockups	
		Test & Evaluation Support	
		Test Facilities	
	System/Program Management		
		Systems Engineering	
		Project Management	
	Data		
		Technical Publications	
		Engineering Data	
		Management Data	
		Support Data	
		Data Depository	

Level 1	Level 2	Level 3	Level 4
	Operational/Site Activation		
		Contractor Technical Support	
		Site	
		Construction	
		Site/Ship/Vehicle Conversion	
		System Assembly Installation & Checkout on Site	
	Common Support Equipment		
		Organizational	
		Intermediate	
		Depot	
	Industrial Facilities		
		Construction/Conversion/Expansion	
		Equipment Acquisition or Modernization	
		Maintenance	
	Initial Spares & Initial Repair Parts		

REFERENCES

1. Boehm, B.W., "Software Engineering," IEEE Transactions on Computers, Volume C-25, Number 12, December 1976, p. 1227.

2. "SLIM System Description," Quantitative Software Management, Inc.

3. "JS-1 Schedule and Cost Estimation System User's Manual," Computer Economics, Inc., 1981.

4. "PRICE Software Model User's Manual," RCA Corporation.

5. Boehm, B.W., Software Engineering Economics, Prentice-Hall, Englewood Cliffs, 1981.

6. Dumas, R.L., "Final Report: Software Acquisition Resource Expenditure(SARE) Data Collection Methodology," The MITRE Corporation, Report Number MTR-9031, December 1983.

7. Apgar, H., "Software Standard Module Estimating," Bunker Ramo Electronic Systems, 1982.

8. Clapp, J.A., "A Review of Software Cost Estimation Methods," The MITRE Corporation, Report Number MTR-3264, August 1976.

9. Aron, J.D., "Estimating Resources for Large Programming Systems," Software Engineering: Concepts and Techniques, edited by J.M. Buxton et al, Litton Educational Publishing, Inc., 1976.

10. Walston, C.E. and Felix, C.P., "A Method of Programming Measurement and Estimation," IBM Systems Journal, Vol. 16, No. 1, 1977.

11. Putnam, L.H., "A General Empirical Solution to the Macro Software Sizing and Estimating Problem," IEEE Transactions on Software Engineering, Vol. SE-4, No. 4, July 1978.

12. Herd, J.R., et al., "Software Cost Estimation Study: Study Results," Doty Associates, Inc.. Report Number RADC-TR-77-220, Vol. I, June 1977 AD A042264

13. Thibodeau, R., "An Evaluation of Software Cost Estimating Models," General Research Corporation, Report Number RADC-TR-81-144, June 1981 AD A104226

REFERENCES (Continued)

14. Wolverton, R.W., "The Cost of Developing Large-Scale Software," IEEE Transactions on Computers, June 1974.

15. Chrysler, E., "Some Basic Determinants of Computer Programming Productivity," Communications of the ACM, Vol. 21, No. 6, June 1978.

16. Sunohara, T., et al., "Program Complexity Measure for Software Development Management," Proc., Fifth International Conference for Software Development Management, IEEE, March 1981, pp. 100-106.

17. Boehm, B.W., et. al., Characteristics of Software Quality, North-Holland Publishing Co., New York, 1978.

18. Musa, J.D., "The Measurement and Management of Software Reliability," Proc. of the IEEE, Vol. 68, No. 9, Sep tember 1980.

19. McCabe, T.J., "A Complexity Measure," IEEE Transactions on Software Engineering, Vol. SE-2, No. 4, 1976, pp. 308-320.

20. Musa, J.D., "Software Reliability Measurement," The Journal of Systems and Software 1, Elvesier North Holland, Inc., New York, 1980.

21. Halstead, M.H., Elements of Software Science, Elvesier North Holland, Inc., New York, 1977.

22. Mathis, N.S. and Willmorth, N.E., "Software Milestone Measurement Study," Naval Electronics Laboratory Center, Report Number NELC-TD-285, November 1973.

23. Brown, J.R., "Impact of MPP on Software Development," TRW Defense and Space Systems Group, Report Number RADC-TR-77-121, May 1977 AD A040467

24. Gilb, T., Software Metrics, Winthrop Publishers, Cambridge, 1977.

25. "Work Breakdown Structures for Defense Materiel Items (MIL-STD-881A)," Department of Defense, 25 April 1975.

REFERENCES (Continued)

26. Tausworthe, R.C., "The Work Breakdown Structure in Software Project Management," The Journal of Systems and Software I, Elvesier North Holland, Inc., 1980, pp. 181-186.

27. "Standard Cost Breakdown Structure," National Security Agency, Draft NSA/CSS Circular (Unnumbered), 1983.

28. Wheaton, M.J., "Software Sizing Task Final Report," The Aerospace Corporation, Technical Memorandum Number ATM-84(45-2303)-1, October 1983.

29. "Proposed Military Standard on Defense System Software Development (MIL-STD-SDS)," Joint Logistics Commanders, 15 April 1982.

30. "Configuration Management Practices for Systems, Equipment, Munitions, and Computer Programs (MIL-STD-483)," Department of Defense.

31. "Weapon System Software Development (MIL-STD-1679A)," Department of Defense.

32. "Specification Practices (MIL-STD-490)," Department of Defense.

33. "Technical Reviews and Audits for Systems, Equipment Munitions, and Computer Programs (MIL-STD-1521A)," Department of Defense.

34. "Proposed Revisions to Data Item Descriptons," Joint Logistics Commanders, Working Papers, 5 December 1983.

35. Najberg, A.C., "Software Data Base Development - Preliminary Report," The Analytic Sciences Corporation, Report Number TIM-4612-5-1, 9 March 1984.

END
JAN.
1988
DTIC